UNIVERSISDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Departamento de Tecnología Electrónica

DISEÑO, IMPLEMENTACIÓN Y MEDIDA EXPERIMENTAL DE CONTROLES EN LAZO CERRADO PARA CONVERTIDORES CONMUTADOS UTILIZANDO PROCESADORES DIGITALES DE SEÑAL DSPIC, ORIENTADO A LA VALIDACIÓN DE UNA HERRAMIENTA CAD DE DISEÑO AUTOMÁTICO DE REGULADORES

Proyecto Fin de Carrera

INGENIERÍA INDUSTRIAL

AUTOR: Carlos Serrano Gómez

TUTOR: Pablo Zumel Vaquero

Octubre 2011

Índice de figuras:

Lista de acrónimos y abreviaturas

I+D+I: Investigación Desarrollo e Innovación

GSEP: Grupo de sistemas electrónicos de potencia

FPGA: Field Programmable Gate Array

DSP: Digital Signal Processor

dsPIC: Digital Signal Processor (Microchip™)

MCU: Micro Controller Unit

CAD: Computer Aided Design

IGBT: Insulated-Gate Bipolar Transistor

MOSFET: Metal Oxide Semiconductor Field Effect Transistor

CA: Corriente Alterna

CC: Corriente Continua

VHDL: Hardware Description Language

CPU: Central Processing Unit

E/S: Entrada/Salida

GPIO: General Propose Input Output

PWM: Pulse Width Modulation

ADC: Analog to Digital Converter

UART: Universal asynchronous receiver/transmitter

ALU: Aritmetic Logic Unit

FDT: Función de Transferencia

FC: Frecuencia de Cruce

MF: Margen de Fase

MCC: Modo de conducción continua

1 Motivación, objetivos y planteamiento del proyecto.

1.1 Contexto, motivación y objetivos del proyecto

Este proyecto fin de carrera se ha desarrollado en el Grupo de Sistemas Electrónicos de Potencia de la Universidad Carlos III de Madrid, dentro dos de sus líneas de trabajo: control digital y elaboración de software de ayuda al diseño de convertidores electrónicos de potencia.

El control digital de convertidores conmutados de potencia se ha utilizado desde hace varias décadas en aplicaciones como el control de motores o grandes sistemas de potencia. Sin embargo, en los últimos años se ha incorporado de manera novedosa a los convertidores de media y pequeña potencia, donde el control analógico era la alternativa principal. A lo largo de la última década se ha desarrollado una amplia actividad de I+D+I en el control digital de convertidores conmutados, donde se han utilizado con éxito FPGA, microcontroladores de bajo coste, DSP, etc. En el GSEP desde hace varios años se viene desarrollando investigación en controles digitales para convertidores conmutados, tanto con FPGA como con plataformas tipo DSP.

Por otro lado, la automatización del diseño y las herramientas CAD específicas para la electrónica de potencia han experimentado un notable desarrollo en los últimos tiempos. En este contexto, el GSEP ha desarrollado en esta línea de trabajo varios programas de ayuda al diseño en electrónica de potencia. Uno de ellos es SmartCrtl (Smart control), desarrollado y comercializado internacionalmente por Powersim Technogies. Este módulo se presenta solo o integrado en el paquete del simulador PSIM, específicamente diseñado para simular circuitos electrónicos de potencia. SmartCtrl es una herramienta de diseño de lazos de control para convertidores conmutados, contemplando una amplísima variedad de convertidores y tipos de control. La edición comercializada hasta ahora proporciona diseño de controles analógicos y se está trabajando en la incorporación del control digital en futuras versiones.

En este contexto es donde surge este proyecto fin de carrera. La motivación principal de este proyecto es la validación experimental del procedimiento de diseño de controladores digitales susceptible de ser incluido en una herramienta CAD como SmartCtrl, así como la detección de posibles mejoras del proceso de diseño. Como punto de partida se ha elegido una plataforma digital basada en un microcontrolador fácilmente accesible por el público (dsPIC de Microchip), así como unos convertidores conmutados también disponibles comercialmente. Esta validación puede ser así reproducida por futuros usuarios.

Los objetivos concretos principales que se plantean son los siguientes:

- Adaptación a la implementación de reguladores digitales lineales de un procesador digital de señal de la familia dsPIC. Incluye la elaboración de los programas y el manejo del entorno de desarrollo.
- Adaptar el proceso de diseño actual de SmartCtrl a controles digitales, con las particularidades del dispositivo digital seleccionado.
- Establecer un procedimiento de medida experimental de lazos de control en el sistema.
- Realizar y documentar una serie de ejemplos y casos que puedan servir como una validación de todo el proceso de diseño y medida.

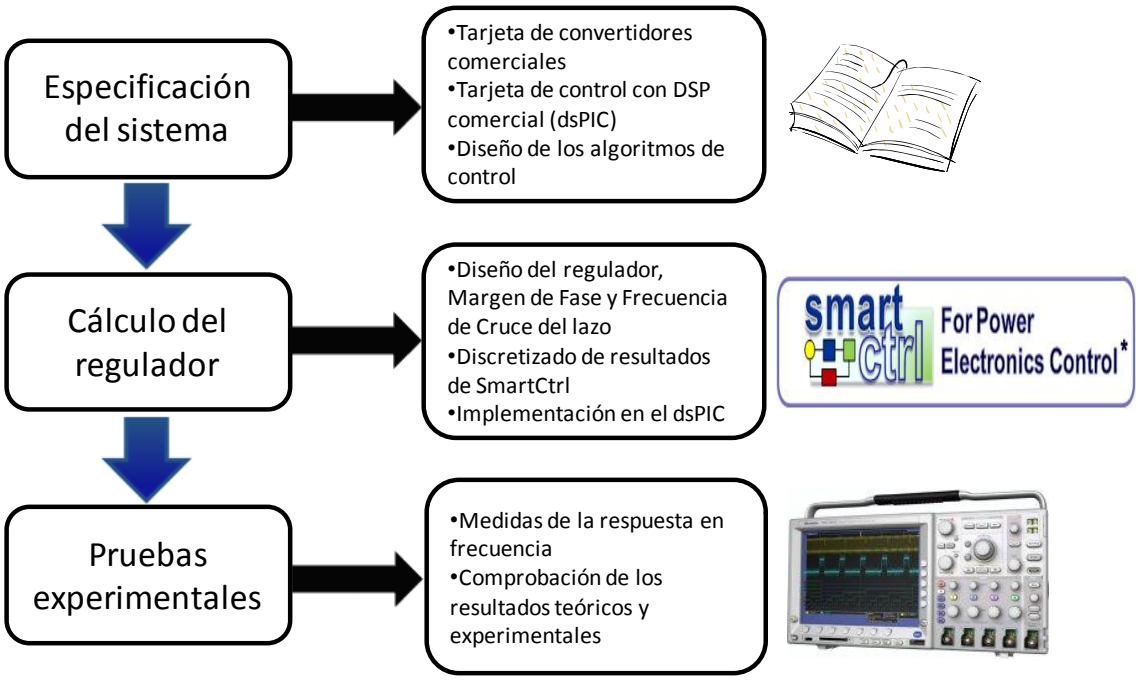


Figura 1-1 Proceso de diseño

1.2 Planteamiento del proyecto

Esta memoria describe el proceso y trabajo realizados a fin de alcanzar los objetivos descritos anteriormente. Se comienza con una introducción a los convertidores conmutados y su control en lazo cerrado en el capítulo 2, comentándose las distintas opciones existentes para lograrlo. Por un lado el control analógico frente al digital, y posteriormente dentro del control digital, mediante FPGA (*Field programable gate array*) o DSP (*Digital Signal Processor*) explicándose que en este proyecto se han desarrollado controles en lazo cerrado mediante control digital por DSP.

A continuación se describe el entorno y prototipo sobre el que se han implementado los distintos controles, donde primero se describe el DSP (capítulo 3) y después las tarjetas de control, potencia y entorno de programación para desarrollar los trabajos de este proyecto (Capítulo 4). Todas estas herramientas son comerciales para permitir que cualquier persona familiarizada con estos temas pueda realizar o ampliar los trabajos descritos sin depender de un complejo proceso de desarrollo y prototipado de tarjetas electrónicas.

En el capítulo 5 se describe cómo realizar los controles implementados en este proyecto basándose en el uso de la herramienta CAD SmartCtrl para diseñar de una manera rápida y visual controles en lazo cerrando. Después mediante MatLab se adapta a un control digital que finalmente es implementado en el DSP mediante el software gratuito que el fabricante de los DSP de este proyecto ofrece para el uso de sus productos. Al final de este apartado se proponen unos pasos a seguir resumidos en una tabla para poder diseñar e implementar controles, que son los mismos que se han seguido en el desarrollo de este proyecto para hacer los reguladores.

En el capítulo 6 se describe y explica cómo se implementan estos controles en el DSP de este proyecto (dsPIC33F de Microchip). Es un proceso complejo que ha requerido mucho

trabajo y dedicación de este proyecto de final de carrera para adaptarse al DSP, su memoria, sus programas, la implementación del algoritmo de control, etc. Sin embargo desde el punto de vista de quien quiera diseñar sus propios controles en esta plataforma, se convierte en trabajo ya realizado que evita al diseñador de reguladores trabajar en este tema, pudiendo simplemente usando lo desarrollado en este proyecto centrarse en aspectos propios del diseño de control.

En el capítulo 7 se describe la metodología y particularidades de este proyecto para realizar las medias experimentales y como se han tomado estas.

Los resultados experimentales están en el apartado 8, donde se exponen 9 casos de reguladores distintos, 3 PI, 3 tipo2 y 3 tipo3, que han sido medidos y detallados en profundidad para validar los objetivos iniciales propuestos en este proyecto.

El proyecto termina con las conclusiones extraídas tras los trabajos realizados haciendo referencia a los objetivos con los que se inició este proyecto de fin de carrera.

2 Introducción a los convertidores conmutados

Los convertidores conmutados son circuitos electrónicos diseñados y dimensionados para realizar conversiones de una forma de energía eléctrica a otra. Estos convertidores se utilizan en aplicaciones cada vez más pequeñas, ligeras y por supuesto más eficientes. En este tipo de circuitos electrónicos existen dos partes claramente diferenciadas, la parte de potencia y la de control.

La parte de potencia es la encargada de transmitir la energía desde la entrada a la salida del convertidor mediante el uso de semiconductores de potencia (IGBT, MOSFET, etc) y componentes pasivos, donde tienen una especial relevancia los componentes que almacenan energía como bobinas y condensadores. Esta capacidad de almacenar energía se usa para poder adecuar la energía de una parte del convertidor, para a la salida tener lo que el equipo a alimentar necesita. En determinados momentos, estos componentes almacenan energía y en otros descargan hacia la carga. Los semiconductores de potencia trabajan en conmutación, es decir, idealmente nunca conviven tensión y corriente. Por lo tanto, con componentes ideales las pérdidas en un convertidor conmutado son cero.

Por otro lado está la parte de control. Esta parte se encarga de controlar y gestionar la cantidad de energía de la que puede disponer el convertidor en la entrada, para ceder más o menos a la salida en función de las necesidades de la carga, siendo este control en tiempo real.

El esquema general del funcionamiento de estos convertidores es el que aparece en la Figura 2-1.

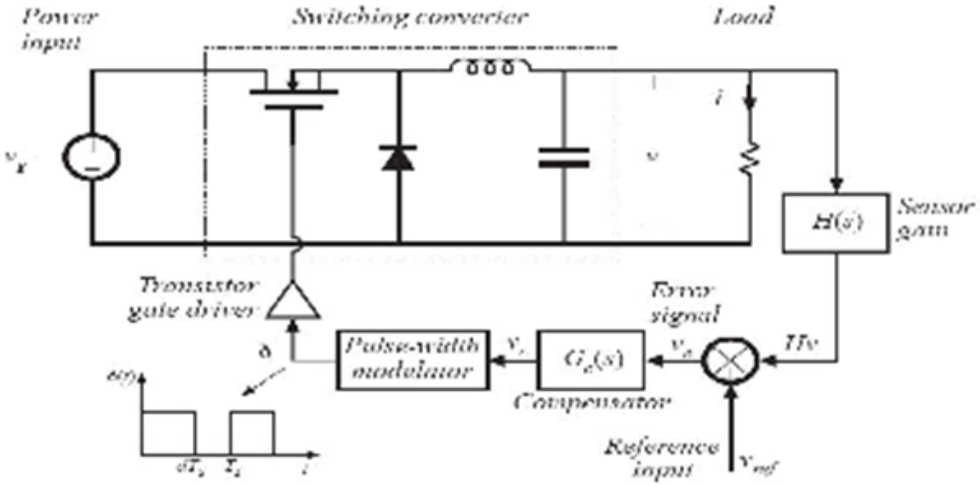


Figura 2-1 Esquema general de convertidor controlado en lazo cerrado, figura extraída de [1]

Como se ve, la parte de control mide el valor de la magnitud física a controlar, tensión o corriente y en función de esa medida actúa para enviar más o menos potencia hacia el punto donde se realizó la medición. Para ello se actúa sobre uno o varios semiconductores de tal modo que los componentes reactivos (bobinas y condensadores) se cargan o descargan en función del estado en el que se encuentre el convertidor, determinado por las señales de disparo de los semiconductores de potencias generadas por la parte de control.

2.1 Descripción general de los tipos de convertidores

Los principales tipos de convertidores son los siguientes:

- Rectificador CA/CA o rectificador
- Conversor CC/CA o inversor
- Convertidor CC/CC
- Reguladores de alterna CA/CA o cicloconvertidores

2.2 Descripción general del control de fuentes conmutadas

Existen diferentes técnicas para el control de estas fuentes conmutadas. Como se dijo anteriormente, la parte de control se encarga de hacer conmutar a los semiconductores del convertidor, para transmitir más o menos potencia a la carga en función del nivel de tensión, corriente o de ambos en la salida e incluso en función de la entrada.

Para la implementación de este control existe una primera gran división entre el control analógico y el digital. La tecnología analógica está más desarrollada y no presenta los problemas de muestreo que si tiene lo digital. En cualquier caso todo depende de la aplicación o sistema donde se quiera integrar el convertidor. En este proyecto se ha trabajado con control digital por lo que no se profundizará más en lo analógico. A continuación se enumeran algunas de las ventajas y desventajas del control digital.

Ventajas:

- Inmunidad al ruido
- Flexibilidad y programabilidad
- Mayor posibilidad de integración
- Permite aprovecharse de la enorme evolución de la tecnología digital

Inconvenientes:

- Necesidad de incorporar un conversor analógico-digital
- Efectos del muestreo de la señal a controlar

Una vez ya centrados en el control digital existe una nueva bifurcación entre el control mediante FPGA o DSP.

2.2.1 Introducción al control digital mediante FPGA

Este tipo de control se está extendiendo dentro del campo de aplicación de las fuentes conmutadas. Se trata de usar una FPGA como controlador del convertidor. Las FPGA, son dispositivos de semiconductores organizados por bloques que permiten implementar una lógica concreta mediante la interconexión de los semiconductores en su interior. Esta lógica es reconfigurable mediante lenguajes de descripción Hardware, siendo el más extendido el VHDL. Esta capacidad de reprogramar hace flexible este sistema, de tal modo que la misma FPGA puede controlar distintos convertidores siempre y cuando esta se reconfigure para la aplicación concreta. Mediante las FPGA se puede implementar una lógica sencilla de una o pocas puertas o un complejo sistema secuencial.

Para el control digital de convertidores conmutados hay que destacar la capacidad de ser concurrente, es decir, ejecución de tareas simultánea. Esto implica que los retardos de cálculo entre las señales de entrada y salida pueden reducirse significativamente respecto de una ejecución de tareas secuencial. Esto constituye su principal característica y ventaja frente a los DSP que se comentarán más adelante. Para su funcionamiento como controlador, se requiere de un conversor analógico-digital para convertir la señal a controlar (tensión o corriente) en un valor digital que pueda interpretar la FPGA, lo cual encarece el sistema y añade algo más de dificultad, aunque para muchas aplicaciones estas dificultades están superadas a un coste competitivo.

2.2.2 Introducción al control digital mediante DSP

Por último existe la opción de implementar el control del convertidor mediante un microcontrolador o DSP (*digital signal processor*). Se trata de circuitos integrados que incluyen en su interior las funcionalidades de una computadora, que son la CPU, la memoria y las unidades de E/S. En el mercado existen multitud de microcontroladores de distintos fabricantes tanto para propósito general como específicos en función de la aplicación, para lo cual disponen de unos u otros periféricos controlados por el DSP. Para el caso de DSP optimizados para el control de fuentes conmutadas es muy útil que incluyan un módulo PWM, conversor analógico-digital, un reloj interno y en caso necesario pueden ser útiles buses de comunicación, pero más bien para integrar el sistema en otra red, que para la propiamente dicha gestión y control de la potencia del convertidor.

Los microcontroladores pueden ser programados con lenguajes de alto nivel como C, gracias a la existencia de compiladores de estos lenguajes a código máquina, pero también puede ser útil hacerlo en ensamblador. Este control a diferencia de la FPGA conlleva un proceso secuencial en el programa, por lo que las instrucciones se ejecutan una por una en un orden específico. Por lo tanto el algoritmo de control se ejecuta de forma secuencial. Esto no tiene por qué suponer un problema, si en el algoritmo que implementa el lazo de control (regulador) siempre se ejecutan las mismas instrucciones y se asegura que el tiempo de actualización de las señales de control es siempre el mismo. Para ello es importante que en el proceso de cálculo del regulador, se tenga la máxima precaución con instrucciones de tipo bifurcaciones condicionales (if-else) o de bucle (for, while), ya que dependiendo de la condición, el programa irá por un camino u otro. Esto puede hacer que el tiempo de cómputo no siempre sea el mismo, lo que introduce retardos en el control, que afectarán a la estabilidad del sistema.

Por todo ello es de gran importancia tener el máximo control de la programación del algoritmo de cálculo del regulador, siendo aconsejable en algunos casos programar en ensamblador, ya que así se tiene el máximo control del flujo del programa y se minimiza el tiempo de ejecución. Esto se debe a que la traducción a código máquina desde C, depende del compilador, por lo que se puede perder ese control tan importante en estas partes del programa.

Como se explicará más adelante, este factor es una diferencia clave frente a la FPGA pero por otro lado, el control mediante DSP suele ser más barato que la FPGA y además incluye todos los periféricos necesarios para hacer el control. En la Figura 2-2 se ve un gráfico que ilustra el tiempo de ejecución en FPGA y DSP en el caso de no tener controlado el flujo de ejecución del algoritmo de control, al haber puesto bucles o bifurcaciones en esta parte tan crítica del programa.

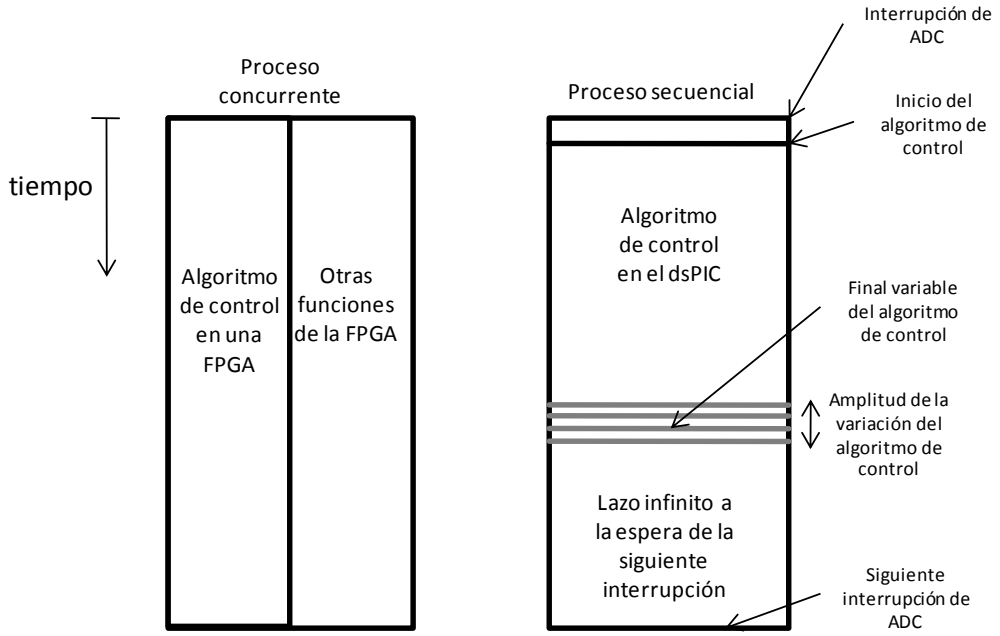


Figura 2-2 Ejecución de proceso concurrente (FPGA) y secuencial (DSP)

Para este proyecto se han usado dsPIC, que son los DSP del fabricante Microchip y algunas de sus demoboards. Los dsPIC usados son el dsPIC33FJ16GS502. Los cálculos del algoritmo de control en lazo cerrado se han hecho siempre en ensamblador, mientras que otras partes del programa (inicialización y programa principal) se han escrito en lenguaje C.

2.3 Descripción general del control mediante dsPIC aplicado a convertidores CC/CC

Este proyecto se ha centrado en el control digital en lazo cerrado de convertidores mediante dsPIC, todo ello con placas comerciales y de fácil acceso para poder realizar y comprobar el funcionamiento de lo hecho en este proyecto así como avanzar en cuestiones no realizadas partiendo de este trabajo.

2.3.1 Introducción al control de convertidores conmutados

Independientemente de que sea control digital o analógico, el objetivo de este es mantener una variable física, en el caso de convertidores conmutados tensión y/o corriente, controladas en tiempo real. El control debe ser capaz de actuar sobre el sistema para corregir desviaciones y absorber o amortiguar perturbaciones que hagan a estas variables alejarse de su valor en régimen permanente. En nuestro caso y de una manera sencilla el lazo de control es el que se muestra en la Figura 2-3.

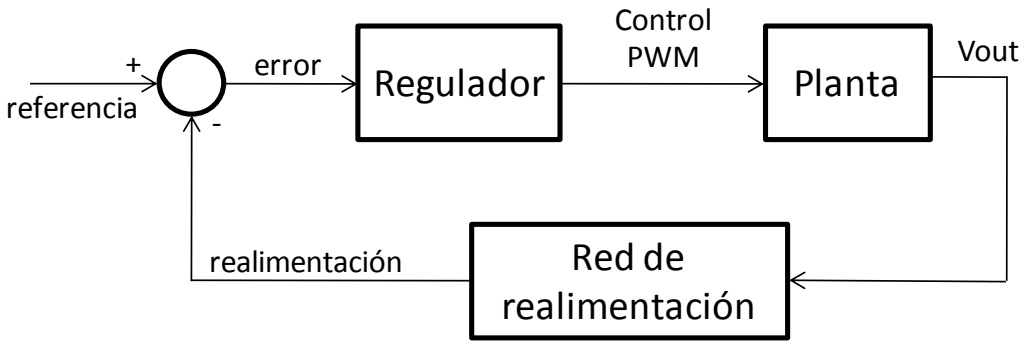


Figura 2-3 Esquema general de un lazo simple de control

En este esquema la planta es el sistema a controlar, es decir un convertidor reductor o un convertidor elevador en este proyecto. La red de realimentación, es la encargada de medir o "sensar" la variable a controlar y adecuarla para la etapa siguiente en el lazo, por ejemplo con un divisor de tensión resistivo. De este modo se adecua la tensión a controlar, por ejemplo la de salida, a los valores de tensión que maneja el controlador. Este valor una vez adecuado al nivel que puede manejar el control, es comparado con una señal de referencia que marcará el nivel a donde se quiere llegar en régimen permanente, es decir, es el valor al cual el regulador intentará llevar el sistema en todo momento y en caso de estar en él, hará lo necesario para mantenerlo. De esta diferencia entre el valor real sensado y el de consigna, se obtiene un error que será mayor cuanto mayor sea la diferencia entre ambos. Esta señal, es la entrada al regulador y en función de este error, actúa para llevar a la planta al valor de referencia. En el caso de convertidores conmutados, se actúa sobre los semiconductores de la planta, reductor y elevador en nuestro caso, para poder transmitir a la salida más o menos energía procedente de la entrada del convertidor. El regulador es en cierto modo un amplificador de error que trata esa señal y en función de la misma la salida será mayor o menor, y una vez modulada tendremos un pulso de control generado por el PWM más o menos ancho.

2.3.2 Respuesta frecuencial de los componentes del lazo

En este proyecto se ha hecho especial hincapié en la respuesta frecuencial del lazo y de los bloques del sistema. Para realizar el control de sistemas, es imprescindible conocer su comportamiento dinámico. Esto quiere decir que cada componente se comporta de diferente manera en función de las perturbaciones o entradas que recibe. Del mismo modo que los condensadores tienen su dinámica y no es posible variar bruscamente su tensión, sino que se produce un comportamiento transitorio, todos los componentes de estos sistemas tienen su dinámica y tardan un tiempo en alcanzar sus régimen permanente. Una manera de conocer el comportamiento dinámico de estos componentes, es conocer su respuesta en frecuencia, o dicho de otra manera su comportamiento ante una variación cíclica, que va variando cada vez más rápido. Esta es una forma rápida y experimental de conocer el comportamiento dinámico de los componentes del sistema. Si se puede conocer cómo reacciona a 1, 10, 100Hz y así sucesivamente hasta 1MHz, es posible ver cómo reacciona ante cualquier variación, ya que se ha medido el resultado que tiene el sistema ante variaciones lentas (1Hz) y también a las rápidas (1MHz). Estas medidas se han realizado en el laboratorio con el analizador de ganancia y fase lo que permitido trabajar toda la parte experimental, ya que se han medido tanto la ganancia como la fase en diagramas de Bode para todos los bloques del lazo.

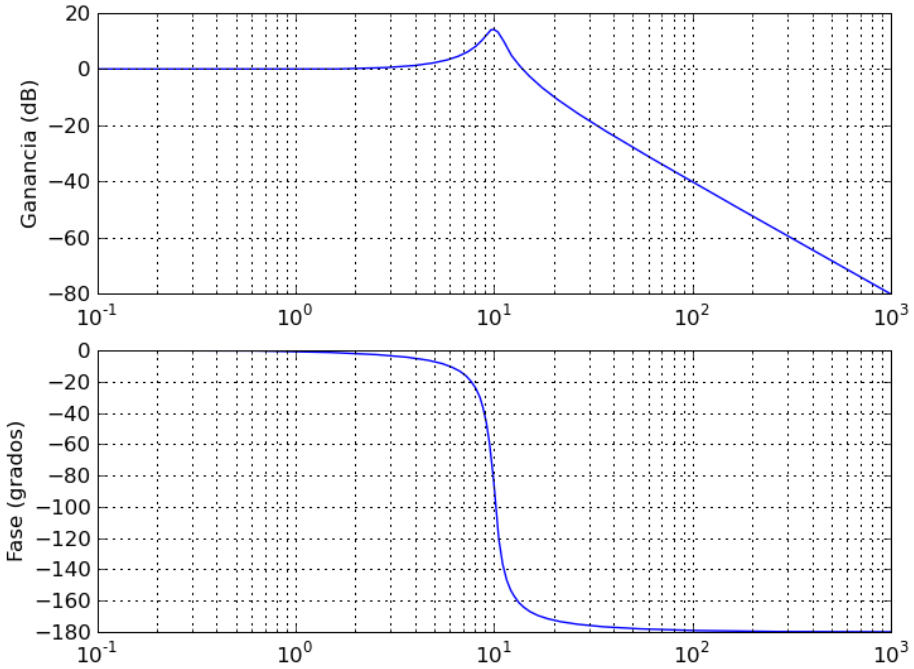


Figura 2-4 Diagrama de bode típico (función de transferencia tensión de salida/ciclo de trabajo en un convertidor reductor ideal)

En este proyecto se han realizado distintos tipos de control sobre un DSP, con distintos algoritmos, y se ha comprobado experimentalmente que se comporta como se esperaba en términos de respuesta en frecuencia del sistema.

2.3.3 El control digital mediante dsPIC

Como se ha dicho anteriormente, se va a trabajar sobre el control digital mediante un dsPIC. Este tipo de control tiene ciertas particularidades que se comentan a lo largo de esta memoria.

La respuesta en frecuencia depende del algoritmo de control, que consiste en el cálculo de una ecuación en diferencias. En el caso de los DSP, son capaces de procesar señales de forma secuencial, es decir, pueden tomar una señal y realizar una operación sobre ella, después otra operación y así hasta que el algoritmo de control esta completo y actualiza la salida del regulador, en este caso el PWM, para después esperar hasta la próxima solicitud del cálculo. Una cuestión muy importante es asegurar que el tiempo transcurrido entre dos actualizaciones consecutivas de la señal de control sea siempre el mismo.

Este problema es solucionable teniendo el control sobre la ejecución del programa haciéndolo en ensamblador.

Este factor de conocer cuánto tarda en calcularse el regulador es de gran importancia, porque afecta a la fase del regulador en forma de retardos y por lo tanto afecta al lazo y a la estabilidad del sistema, por lo que es necesario conocerlo. En las siguientes figuras se muestran, dónde y cuánto tarda el realizarse el cálculo, en actualizarse el PWM y que problema supone no tener esto controlado. En la primera de estas figuras, se ve un caso

favorable en el que la actualización del PWM, se hace tras haber realizado el cálculo del regulador tomando los datos de muestras del instante anterior.

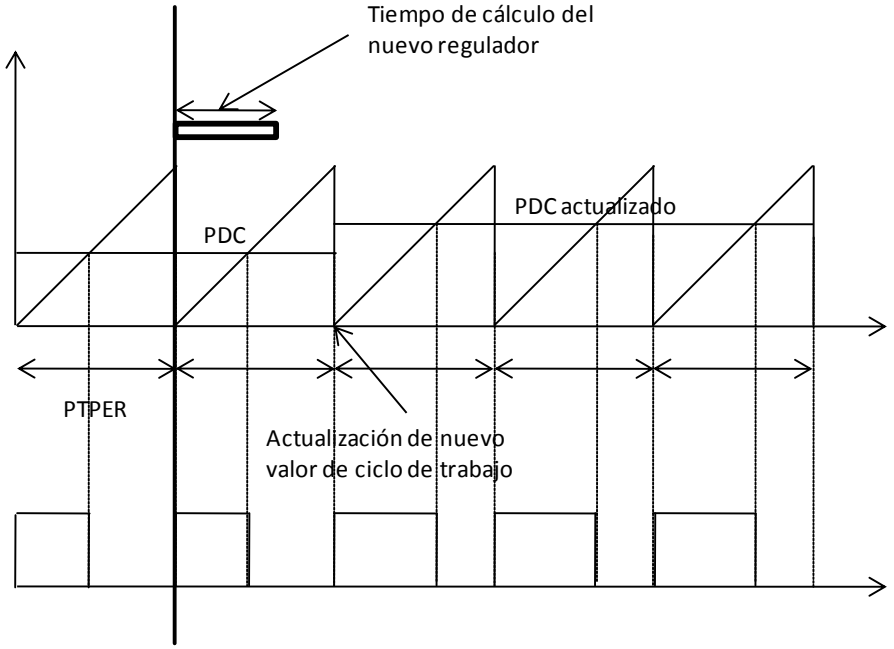


Figura 2-5 Diagrama de actualización del PDC con retardo sencillo

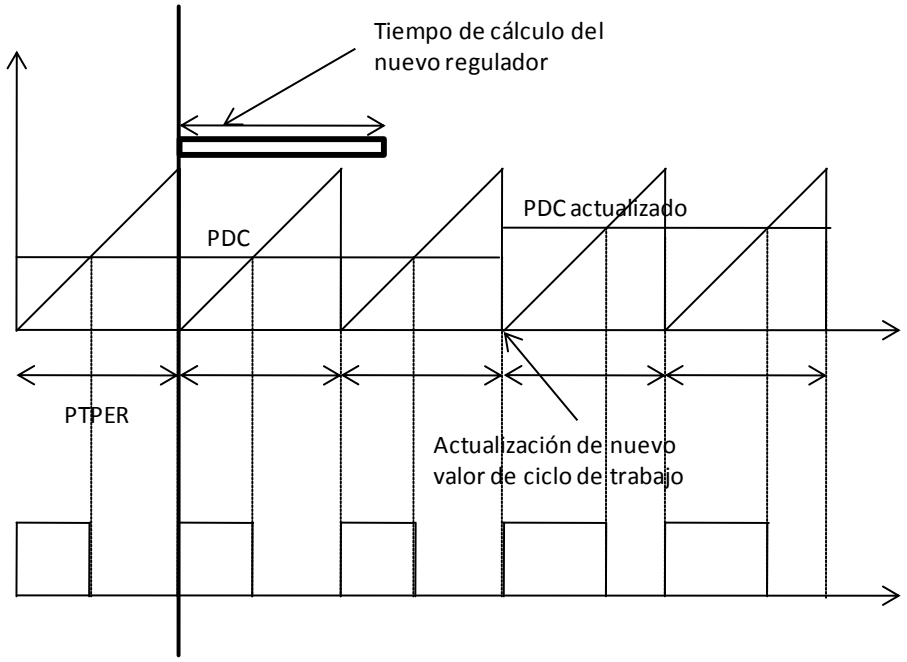


Figura 2-6 Diagrama de actualización del PDC con retardo doble

Sin embargo en el segundo gráfico vemos que los cálculos tardan más de un periodo de PWM, y eso hace que al actualizarse el PWM, se haga con unos cálculos que tomaron datos de error y ciclo de trabajo en dos muestras anteriores, lo que supone unos retardos que afectan a la fase del regulador, con ello a la del lazo y por lo tanto a la estabilidad del sistema. Por lo tanto esta situación debe ser estudiada con especial detenimiento. Si además de este problema sumamos que unas veces es un caso y luego otro, se pierde aún más el control de

estos retardos y de la fase. Por lo tanto, el objetivo es conseguir que el funcionamiento se asemeje al de la Figura 2-6.

Por otro lado el control mediante DSP, tiene las características propias de todo control digital al igual que el control mediante FPGA. La respuesta en frecuencia del regulador se puede calcular mediante las mismas técnicas que el analógico, pero una vez obtenida la función de transferencia (FDT) del regulador en continuo (espacio S) este es necesario discretizarlo ya que se va a trabajar con regulador en discreto (espacio Z). O bien es posible calcular el regulador directamente en discreto. En este proyecto se ha aprovechado la herramienta SmartCtrl de PSIM, para hacer el diseño del regulador y después se ha discretizado usando MatLab.

Por lo tanto, tras el proceso de diseño del regulador con SmartCtrl, se obtiene una FDT en continuo de la forma:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + \cdots + b_{m-1} s + b_m}{a_0 s^n + \cdots + a_{n-1} s + a_n} \qquad \textit{Ec. 2-1}$$

Donde para este proyecto el regulador de mayor orden será el tipo 3 con a_0=1

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^3 + b_1 s^2 + b_2 s + b_3}{a_0 s^3 + a_1 s^2 + a_2 s + a_3} \qquad \textit{Ec. 2-2}$$

A partir de esta FDT, se dicretizará para obtener una en discreto de la forma:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 z^3 + b_0 z^2 + b_2 z + b_3}{a_0 z^3 + a_1 z^2 + a_2 z + a_3} \qquad \textit{Ec. 2-3}$$

Que multiplicando arriba y abajo por el Z^n de mayor exponente y con a_0=1 queda:

$$G(z) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} \qquad \textit{Ec. 2-4}$$

Esta sería la FDT discreta del regulador a implementar. Que pasándolo a ecuación en diferencias queda:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} + b_3 \cdot e_{n-3} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} - a_3 \cdot d_{n-3}$$

Ec. 2-5

Donde e_n es el error actual (diferencia entre la referencia y el valor medido de la variable de control), e_{n-1} el error en el instante de cómputo anterior y así sucesivamente hasta e_{n-m}, que depende del tipo de regulador. Por otro lado, d_n es la señal de control, que en este caso es el ciclo de trabajo que va a actuar en el PWM, d_{n-1} es el valor del ciclo de trabajo en el instante anterior y así de nuevo sucesivamente hasta d_{n-p}, que depende del tipo de regulador. Al final queda una ecuación como la Ec. 2-5, que es la que implementa y procesa el dsPIC, tomando los valores de error y ciclo de trabajo de distintas posiciones de memoria, realiza el cálculo y actualiza esas mismas zonas de memoria, con los nuevos valores. Ese es el algoritmo de control, y es sobre este sobre el que se han realizado los cambios para tener uno u otro regulador.

De aquí podemos ver de nuevo porque es tan importante conocer cuándo se realiza la toma de muestras, cuánto tarda en calcularse el control y en actualizarse el ciclo de trabajo y conocer si esto puede hacerse en el ciclo de PWM. El proceso de toma de datos debe hacerse siempre en el mismo punto, el tiempo de cómputo de la nueva señal de control debe ser

siempre el mismo y por último el punto de actualización del ciclo de trabajo debe ser el mismo, quedando todo esto dentro de un periodo de PWM, para que los nuevos cálculos vengan de los valores de error y ciclo de trabajo del instante anterior, y de 2 o más instantes anteriores para que la fase del regulador tenga las mejores prestaciones de cara a la estabilidad del lazo.

Para conseguir que el cálculo del regulador se haga siempre en el mismo punto, se usa una interrupción del ADC que precisamente es provocada por un evento en el PWM. Con esto se consigue que estos cálculos se inicien siempre con eventos del PWM en el mismo punto del ciclo. Para ello es necesario configurar tanto los registros del PWM para que generen este disparo, como los del ADC, para que actúe su interrupción provocada con el PWM, lo que sincroniza ambos y por último hay que configurar registros del administrador de interrupciones para que la del ADC provoque el consecuente salto el flujo normal del programa para poder realizarse el cálculo del nuevo valor de control. Todo esto se explicará en el capítulo siguiente que trata específicamente del manejo del dsPIC.

3 Descripción del dsPIC desde el punto de vista del control de convertidores conmutados

Tanto Microchip como otros fabricantes (Texas Instruments, Freescale, etc) han desarrollado productos específicos para su uso como controladores de convertidores de potencia y han hecho un gran esfuerzo para adecuar las características de sus DSP, a las demandas concretas del control de este tipo de dispositivos. Ejemplos claros del esfuerzo de los fabricantes en este sentido son las familias de dsPIC33 y dsPIC30 de Microchip. Existen productos similares de otras marcas, pero se han elegido estos por la cantidad de información que Microchip ofrece en este mismo sentido lo que da una idea del interés que está mostrando la industria en el campo del control digital. Para conseguir focalizar su interés en estas aplicaciones, los fabricantes añaden periféricos específicos, en la medida de lo posible fáciles de manejar, a sus DSP para poder especializarse en el control digital de convertidores. Para el control digital en sí, tienen especial importancia el ADC y el módulo PWM, que para nuestro dsPIC serán detallados más adelante. Es de especial importancia para conseguir un funcionamiento equiparable al control analógico, que las lecturas del ADC y la velocidad del PWM sean altas.

Cuando se trata de tener una magnitud controlada en un convertidor, es imprescindible conocer en todo momento el valor de esa magnitud, ya sea corriente o tensión, con una precisión suficiente y con capacidad para detectar cambios bruscos en ellas. Por lo tanto el ADC es un elemento de gran importancia. Del mismo modo, si se producen alteraciones en el sistema, es necesario actuar sobre los semiconductores del convertidor con rapidez y precisión para adecuar el ciclo de trabajo a las condiciones en las que se encuentre, función de la que se encarga el periférico PWM.

A parte de estos periféricos, los DSP utilizados incluyen otros no tan importantes para el control o el funcionamiento del convertidor en sí mismo, pero útiles para integrar el convertidor en otros sistemas más complejos o interactuar sobre ellos. Estos podrían ser las E/S digitales, timers, comparadores, UART, periféricos de comunicaciones (I^2C) etc.

Aparte de todos estos periféricos también es recomendable un motor DSP que permita la ejecución suficientemente rápida de un algoritmo de compensación. Este algoritmo puede ser un PI, PID de tipo 2, tipo 3 etc.

En cualquier caso, el cálculo del algoritmo debe ser rápido y además es importante que el tiempo usado en el cálculo sea siempre el mismo, por lo que es recomendable tener total dominio del programa y su ejecución. Por estas razones de rapidez y control de la ejecución se recomienda el uso de ensamblador para realizar este código.

Con carácter general un DSP está diseñado para realizar las tareas más habituales para el procesado de señales digitales como sumas, multiplicaciones y retrasos. Estos disponen de arquitectura Harvard que se caracteriza por tener físicamente separados los bloques de memoria de datos y programas, cuyos direccionamientos son también mediante buses separados, como se puede apreciar en el diagrama de la Figura 3-1.

3.1 Esquema general del dsPIC33FJ16GS502.

Finalmente para este proyecto se ha optado por el dsPIC33FJ16GS502 de Microchip, por sus características y periféricos. En este caso interesó especialmente el número de pines (28), ya que se implementará en una demoboard compatible, los 8 canales PWM y el ADC integrado con opción de disparo desde el PWM.

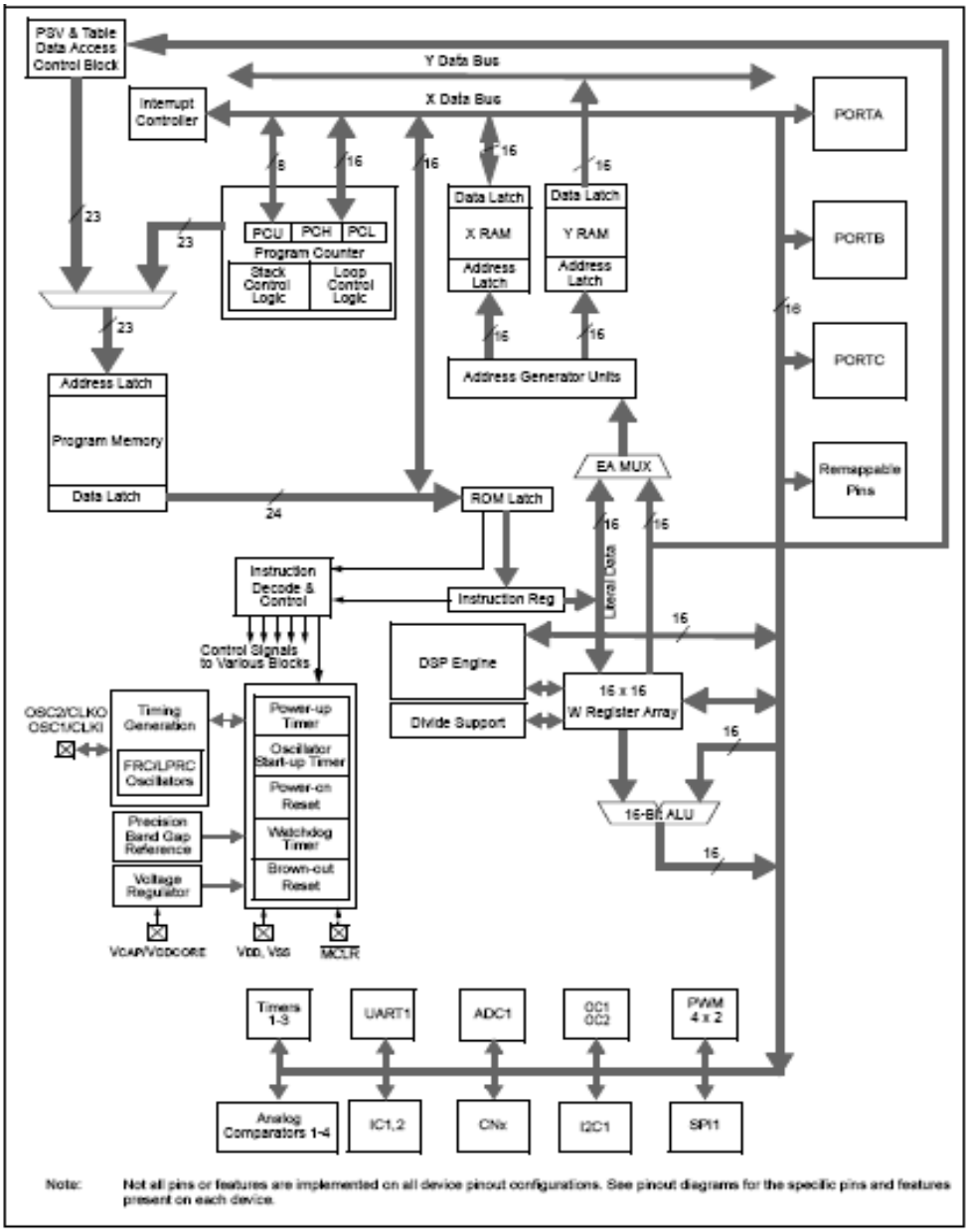


Figura 3-1 Diagrama de bloques general del dsPIC33FJ16GS502, imagen obtenida de [4]

Como se ha comentado anteriormente, los elementos más importantes para realizar un control digital de un convertidor conmutado es el motor DSP, el ADC y el módulo PWM. Por lo tanto, en este proyecto se va a centrar el interés en estas partes. Además este dsPIC ha sido elegido precisamente por estas características ya que son las que lo hacen específico para esta aplicación. Sobre los demás periféricos se comentarán aspectos generales

3.1.1 La CPU del dsPIC33F

La familiadsPIC33F es un DSP de 16 bits con arquitectura Harvard y con su propio set específico de instrucciones de tipo RSIC, tanto las normales MCU, como las propias de un DSP, ambas integradas en una única CPU. Dispone de 24 bits para las palabras de instrucción y 23 bits para el Program Counter (PC), que es el contador que direcciona sobre la memoria de programa, siendo este el que organiza la secuencia y el flujo que sigue el programa. El programador dispone de 16 registros de trabajo de 16 bits (W0-W15) que pueden ser usados para almacenar datos o direccionar de forma directa o indirecta siendo el último (W15) el que puede funcionar como stack pointer en interrupciones y llamadas, almacenando en él, el valor que tenía el PC antes de la interrupción o llamada. Además se dispone también de dos acumuladores de 40 bits.

Como se ha dicho anteriormente existen instrucciones MCU, que son las normales en cualquier microcontrolador como las de movimiento (MOV), saltos o por ejemplo BSET (operaciones a nivel de bit) y existen también instrucciones de tipo DSP, como por ejemplo las de multiplicación-acumulación (MAC). Estas son las que hacen de aquellos microcontroladores que disponen de ellas, específicos para el procesado de señales digitales donde sumas, multiplicaciones o retrasos son muy usados y deben ser ejecutados con la mayor velocidad posible para que sean competitivos.

La mayoría de las instrucciones DSP son capaces de ejecutar la lectura de un dato de la memoria, hacer la lectura de un dato de un registro de trabajo, realizar una escritura en memoria de datos y leer y ejecutar una instrucción de la memoria de programa, haciéndose todo ello en un ciclo de instrucción. Esto permite realizar operaciones de tipo A+B=C en un solo ciclo, consiguiendo realizar estas operaciones DSP a alta velocidad, lo cual es precisamente lo que hace de estos dispositivos idóneos para el procesado digital de señales.

El motor del DSP es capaz de realizar multiplicaciones de 17 bits por 17 bits. Y la ALU es capaz de realizar operaciones aritmético-lógicas de 16 bits con los dos acumuladores de 40 bits de los que dispone el DSP. Algunas de estas operaciones de tipo DSP, pueden funcionar en paralelo con otras de tipo MCU lo que permite optimizar las operaciones en tiempo real. Esto permite realizar operaciones de multiplicación, donde se extraen de memoria dos datos, se multiplican y se añaden a lo que haya en el acumulador (con o sin saturación), siendo estas operaciones DSP y al mismo tiempo prepara los registros de trabajo para que apunten a posiciones distintas de memoria y así realizar una segunda operación después, siendo estas últimas operaciones MCU, todo ello en el mismo ciclo. Estas instrucciones requieren de una RAM con los espacios de memoria separados para operaciones DSP o MCU que se explicarán más adelante.

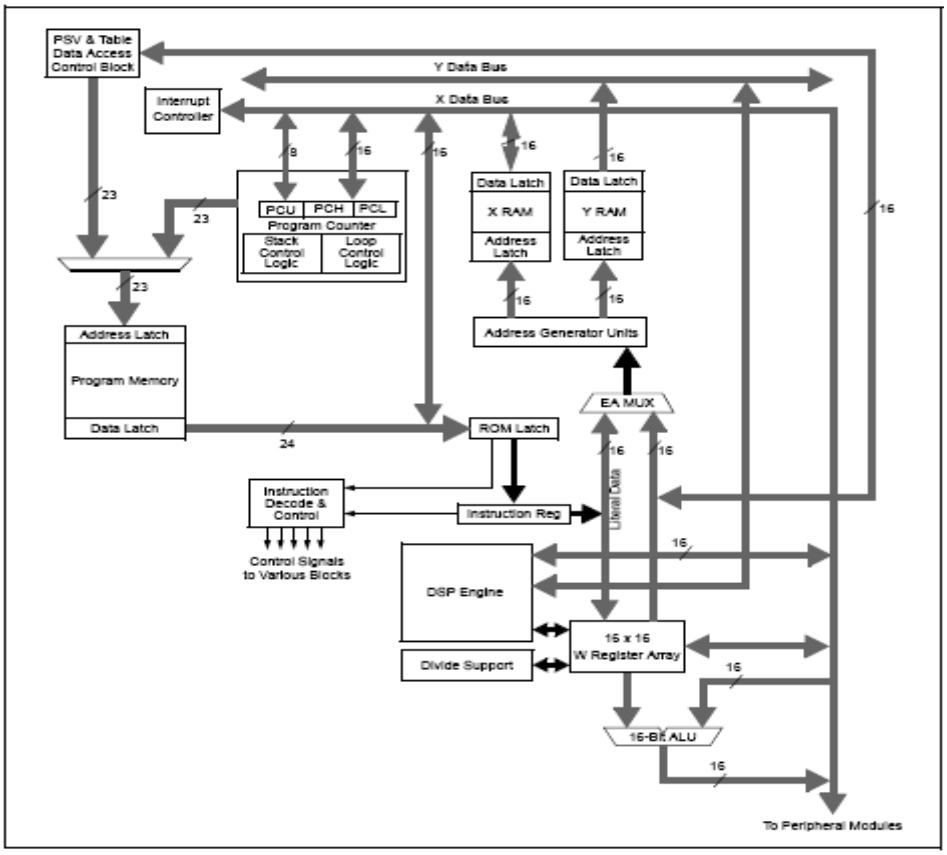


Figura 3-2 CPU core, imagen obtenida de [4]

La ALU es capaz de realizar sumas, restas y desplazamientos de bits lógicos. Puede realizar estas operaciones tanto con 8 como con 16 bits, dependiendo de la instrucción usada y lo datos pueden provenir de un registro de trabajo o de la memoria de datos, dependiendo del modo de direccionado de la instrucción. Del mismo modo el resultado puede almacenarse en un registro de trabajo o en la memoria de datos.

Todas estas prestaciones del dsPIC33F nos permiten realizar el algoritmo del regulador. Aparte de lecturas del ADC y actuaciones en el convertidor a través del PWM, el núcleo de todo es el cálculo de regulador digital, y esto se hace en la CPU. Después, el pulso de control de los interruptores del convertidor será generado con mayor o menor facilidad por el módulo PWM pero sin esta capacidad del dsPIC para realizar sumas, multiplicaciones y divisiones (operaciones MAC), es muy difícil implementar el regulador en el DSP. A continuación se muestra la Tabla 3-1, con la lista de operaciones de tipo MAC, que pueden realizar los DSP para algoritmos de tratamiento digital de señales y un esquema del motor del DSP que realiza estas operaciones en la Figura 3-3.

Instruction	Algebraic Operation	ACC Write Back
CLR	$A = 0$	Yes
ED	$A = (x-y)2$	No
EDAC	$A = A + (x-y)2$	No
MAC	$A = A + (x*y)$	Yes
MAC	$A = A + x2$	No
MOVSAC	No change in A	Yes
MPY	$A = x*y$	No
MPY	$A = x2$	No
MPY.N	$A = -x*y$	No
MSC	$A = A - x*y$	Yes

Tabla 3-1 Instrucciones de tipo DSP, imagen obtenida de [4]

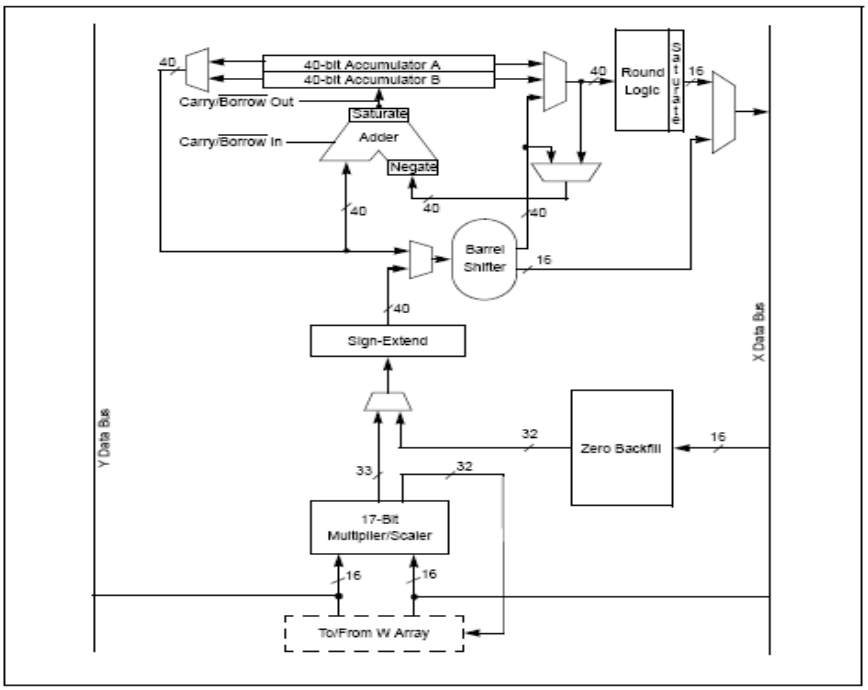


Figura 3-3 Motor DSP, imagen obtenida de [4]

3.1.2 La memoria del dsPIC33F

Es este apartado sólo se comentarán aspectos de la memoria que tengan especial sentido con las necesidades del DSP, para el cálculo de reguladores en aplicaciones de electrónica de potencia u otros aspectos generales.

Como se comentó anteriormente, para los DSP se usa arquitectura Harvard que se caracteriza especialmente por la separación de la memoria de datos y la memoria de programa. En el espacio de memoria de programa, se pueden llegar aproximadamente a 4Minstrucciones (24 bits cada una) siendo este espacio direccionado por los 23 bits del program counter (PC). Como ya se ha comentado, el PC marca la secuencia a seguir en la ejecución del programa así como los saltos provocados por instrucciones o por interrupciones, lo cual no difiere mucho de otros microcontroladores, ni es algo especifico para el control de convertidores, por lo que no se profundizará más en este asunto.

Respecto a la memoria de programa, si tiene un poco más de interés, ya que como se comentó en el apartado anterior, esta memoria es de nuevo dividida en dos espacios, el X y el Y:

El espacio X puede ser usado por todas las instrucciones y mediante todos los modos de direccionamiento. El dsPIC33 dispone de buses distintos para lectura y escritura, siendo el bus de lectura el mismo que se usa para leer las instrucciones y que puede recorrer tanto el espacio X como el Y. Este mismo bus es el que permite realizar operaciones duales DSP y MCU.

El espacio Y, se usa siempre conjuntamente con el X para las operaciones de tipo MAC (Tabla 3-1).

Los límites entre la zona X e Y no son programables por el usuario y dependen del dsPIC a usar. En la Figura 3-4 se muestra un esquema de la ordenación de la memoria.

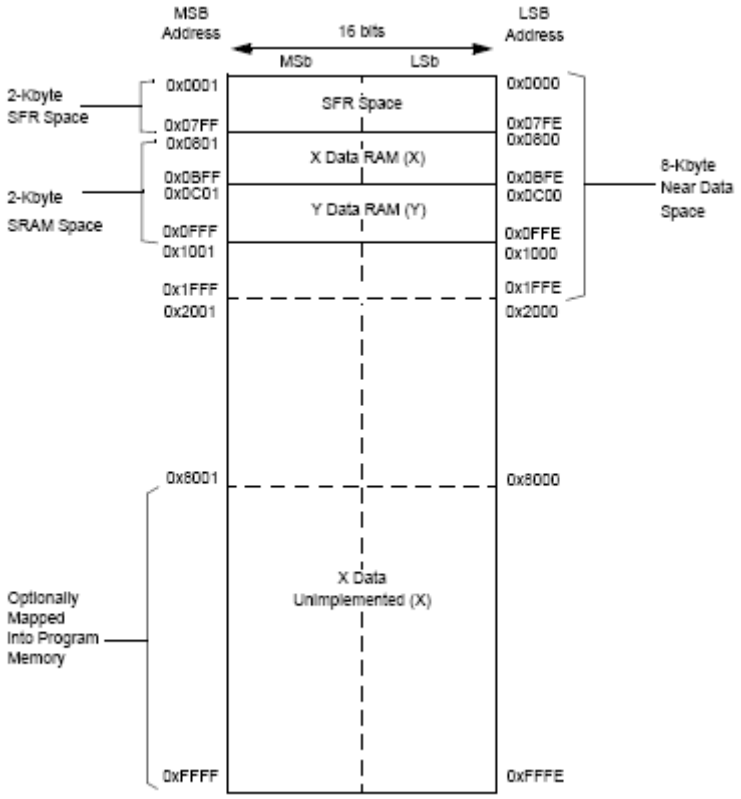


Figura 3-4 Mapa de memoria de datos, imagen obtenida de [4]

3.1.3 El PWM de alta velocidad del dsPIC33F

El periférico para la gestión de PWM, es específico de este DSP y está especialmente diseñado para su uso en aplicaciones de conversión de energía eléctrica, AC/DC, DC/DC, PFC etc. El fabricante proporciona una amplia lista de característica y prestaciones, pero sólo se comentarán algunas de ellas y se profundizará más en aquellas que han tenido una especial relevancia en el proyecto o el funcionamiento del PWM de cara a la aplicación experimental. En este caso, el dsPIC33F dispone de 4 generadores PWM con 8 salidas, 4 “H” y 4 “L”. Cada uno de estos generadores tiene su propia base de tiempos, aunque para este proyecto todos han usado la misma. Dispone de distintos modos de funcionamiento, “complementario”, “independiente”, “push-pull”, “multifase” y otros más, pero los dos únicos que han sido usados son el complementario para un reductor síncrono y el independiente para el elevador, y serán explicados más adelante en el apartado 3.1.3.2. Una de las opciones más importantes de este módulo PWM, es el disparo sincronizado con las lecturas del ADC, ya que es especialmente importante para dar al convertidor buenas prestaciones. La resolución del PWM es de 1.04ns y es aplicable tanto al ancho del ciclo de trabajo, como para los tiempos muertos, los desplazamientos de fase etc.

Dentro de la electrónica de potencia, hay que recordar que el PWM consiste simplemente en pulsos cuadrados de ancho variable que actúan sobre los interruptores del convertidor y determinan la energía que se transmite desde la entrada a la salida.

En la Figura 3-5 se ve el esquema general del módulo PWM, donde se ven los 4 módulos que generan los PWM independientes, registros comunes a todos ellos y todas las funcionalidades de este módulo PWM.

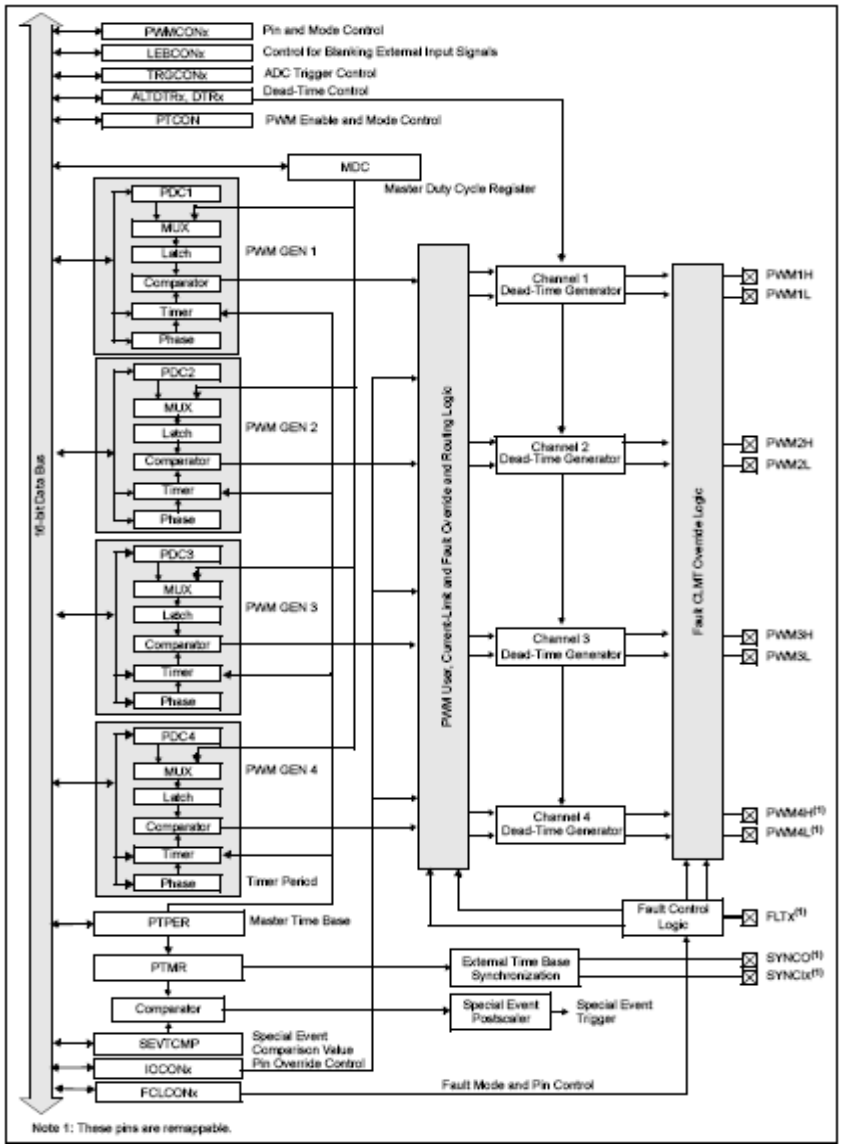


Figura 3-5 Diagrama de bloques del generador PWM, imagen obtenida de [4]

El funcionamiento es sencillo en cuanto a cómo generar un PWM. Lo complicado es configurarlo para la aplicación concreta, ya que este módulo tiene muchas opciones de funcionamiento, por ejemplo cómo generar las rampas, que tipo de PWM se quiere, cuándo dead-time se necesita y si debe ser positivo o negativo. Así hay infinidad de opciones que dependen de la aplicación que se quiera implementar. Reductores, reductores multifase, elevadores, UPS, PFC, full-bridge, y otras aplicaciones requieren modulaciones y detalles muy concretos de cómo generar el PWM y por lo tanto se hará sólo énfasis en las funcionalidades que han tenido relevancia en el desarrollo del proyecto.

3.1.3.1 Generación del PWM

Para construir el PWM, el dsPIC genera una base de tiempos en forma de rampa de subida o contador, a partir del registro PTMR que recibe directamente el reloj interno del DSP. En base a este, el periodo del PWM en este proyecto se fija a través del registro PTPER, que es un registro que delimita el valor al cual la rampa o contador se detiene e inicia otra rampa. El PTPER es configurable por el programador, por lo que el PWM puede tener la frecuencia configurable y con ello definir la frecuencia de conmutación del convertidor. Además del PTPER para definir esta frecuencia de conmutación hay otros métodos más complejos para definir la frecuencia del PWM especialmente pensados para convertidores multifase, pero en este proyecto no han sido usados. En estas condiciones, el programador fija un valor de 16 bits, que especifica el valor de la cuenta del timer PTMR a la que la rampa que genera el PWM finaliza, e inicia de nuevo otra rampa. Esta cuenta que genera el periodo, se activa a través del bit PTEN del registro PTCON que a efectos prácticos supone la activación del todo el PWM.

Para terminar de definir el PWM, se usa el registro MDC para tener todos los generadores iguales o los registros PDCx, para fijar de forma independiente los 4 bloques PWM. En este proyecto se ha usado el PDC1, PDC2 y PDC3 ya que son independientes y para convertidores distintos. Es decir, el PDC se usa para definir el ancho del pulso o el ciclo de trabajo para cada PWM de forma individual, fijando el umbral respecto a la rampa de subida de PTPER para que la señal PWM cambie de estado.

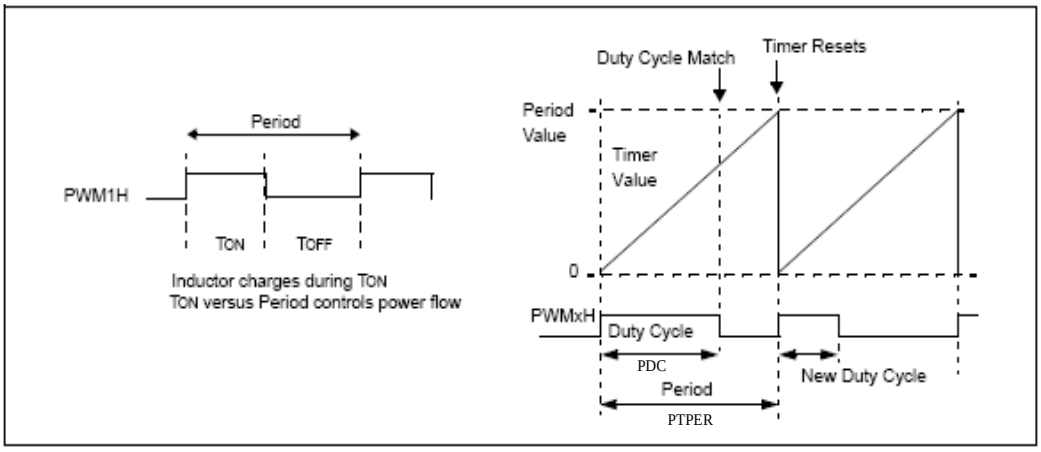


Figura 3-6 Esquema general de generación del PWM, imagen obtenida de [4]

En este proyecto se ha usado este tipo de modulación donde hay rampa de subida y vuelta a empezar, aunque el dsPIC permite otras configuraciones con rampa de subida y bajada doblándose el periodo de conmutación.

3.1.3.2 Modos de funcionamiento

El dsPIC33F permite varios modos de funcionamiento en función de la aplicación para la que va a ser usado. Las más significativas son el push-pull, el complementario, redundante y el independiente. Para este proyecto solo se ha usado el modo complementario para los reductores síncronos y el modo independiente para el elevador.

Para el complementario, el módulo PWM se encarga partiendo de la configuración general del PWM, de construir las señales PWMxH y PWMxL que se conectarán a la puerta los MOSFET a través de un driver o circuito de disparo

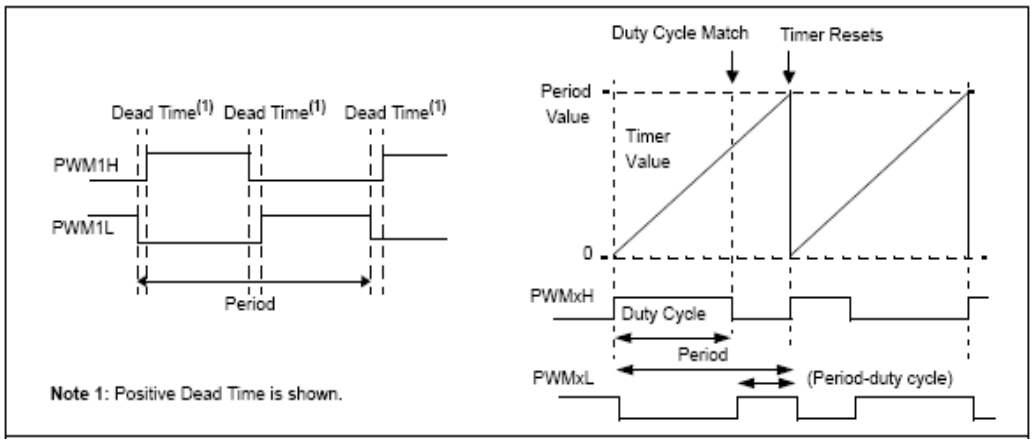


Figura 3-7 Esquema de generación del PWM complementario, imagen obtenida de [4]

En esta configuración de PWM (Figura 3-7), el H es el complementario del L excluyendo la acción del tiempo muerto (dead-time). El pulso H es el que tiene una anchura directamente proporcional al ciclo de trabajo. Tanto en la Figura 3-7 como en el caso real el dead-time es positivo, lo que es estrictamente necesario en un reductor síncrono para que en ningún momento los dos semiconductores del reductor conduzcan. La anchura de este dead-time no ha sido objeto de estudio y simplemente ha sido usado un ancho tal que se garantizase el correcto funcionamiento del reductor. Sin embargo existen también otras configuraciones de dead-time útiles para otras aplicaciones.

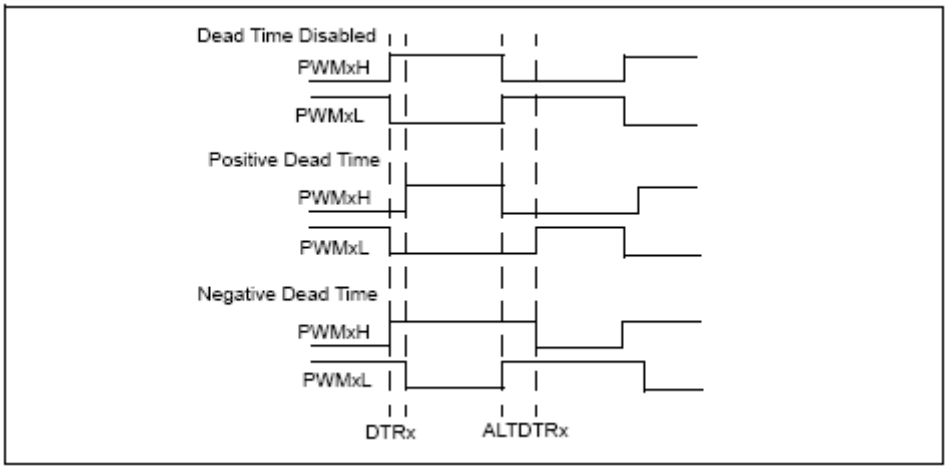


Figura 3-8 Configuración posible del Dead-Time, imagen obtenida de [4]

Para el caso del elevador, el generador PWM ha sido configurado como independiente donde el PWMxH y el PWMxL pueden tener ciclos de trabajo diferentes aunque para nuestro caso solo se ha usado el H, ya que el elevador sólo tiene un semiconductor controlado. En estas condiciones el PDC define el ciclo de trabajo de la señal H y el de la señal L se define con el SDC, el cual no ha sido usado para hacer el reductor y elevador más parecidos.

Por último como se ha dicho anteriormente existen otros métodos que no han sido usados para este proyecto pero que son muy útiles para aplicaciones futuras. El mismo fabricante nos ofrece una tabla (Tabla 3-2) donde se ven aplicaciones de potencia concretas y el modo de funcionamiento del PWM recomendado.

Item	Topology(1)	Configuration	PMOD<1:0> Setting
1	Flyback Converter	True Independent Output mode/Redundant Output mode	11/01
2	Boost/PFC Converter	True Independent Output mode/Redundant Output mode	11/01
3	Interleaved PFC Converter	True Independent Output mode with Master Time Base	11
4	Forward Converter	True Independent Output mode/Redundant Output mode	11/01
5	Double Ended Forward Converter	True Independent Output mode/Redundant Output mode	11/01
6	Active Clamp Forward Converter	Complementary PWM Output mode	00
7	LLC Half-Bridge Series Converter	Complementary PWM Output mode	00
8	Half-Bridge Converter	Push-Pull PWM Output mode	10
9	Push-Pull Converter	Push-Pull PWM Output mode	10
10	Full-Bridge Converter	Push-Pull PWM Output mode	10
11	Phase Shifted Full-Bridge Converter	Complementary PWM Output mode	00
12	Single-Phase Synchronous Buck Regulator	Complementary PWM Output mode	00
13	Multi-Phase Synchronous Buck Regulator	Complementary PWM Output Mode with Master Time Base and Phase Staggering between each Buck converter PWM gate drives	00

Tabla 3-2 Resumen de topologías y su configuración PWM recomendada

3.1.4 El conversor analógico digital del dsPIC33F (ADC)

Para las aplicaciones de electrónica de potencia, es muy importante el tiempo o retardo que se produce entre la solicitud de la toma de medida y el resultado de la misma, siendo también importante en que instante se solicita la medida dentro de un ciclo de conmutación del convertidor. Por lo que tanto las prestaciones del conversor ADC como el instante en el que el programa solicita la conversión son de gran importancia. Para las aplicaciones de potencia que requieren lazos de control de alta frecuencia, es imprescindible que el ADC y el PWM trabajen sincronizados y a gran velocidad. El ADC del dsPIC33F es de 10 bits de resolución y según el fabricante puede realizar conversiones en 0.5 µs, lo cual es bueno para reducir el retardo entre la medida y la actuación del control.

El funcionamiento general del ADC está organizado por pares (AN0-AN1,AN2-AN3…) y cada uno de ellos puede ser utilizado para hacer la conversión mediante 16 fuentes distintas, es decir, que hay hasta 16 motivos por los que se inicia la conversión. Estas fuentes que solicitan la lectura y conversión pueden ser por ejemplo los PWM, un timer que se desborda, por software etc. Esta capacidad resulta de especial utilidad ya que es la que permite sincronizar el ADC con el PWM.

El esquema general del funcionamiento del ADC se muestra a continuación, en la Figura 3-9.

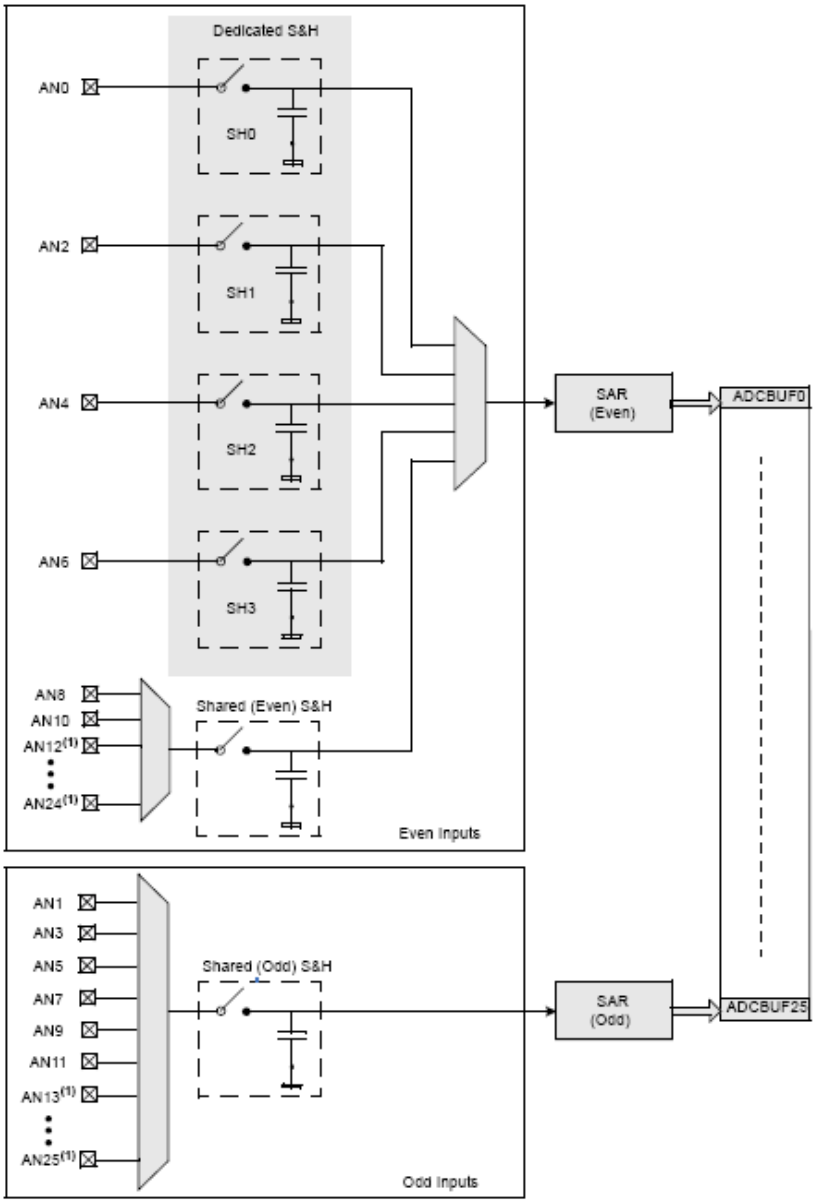


Figura 3-9 Esquema general de funcionamiento del ADC, imagen obtenida de [4]

En la figura se ve que en cada instante solo se realiza la conversión de uno de los pares que es seleccionado por el multiplexor, pero la conversión entre el par y el impar si pueden realizarse a la vez. Si se explica el proceso por pasos queda de la siguiente manera:

- Primero se selecciona con el multiplexor un par para realizar la conversión. Esta entrada analógica se conecta con un circuito llamado *sample&hold* (S&H), que toma el valor analógico de tensión mediante un condensador que se carga hasta el valor analógico de la entrada.

- A continuación se desconecta el circuito de (S&H) dejando el condensador en abierto y cargado.

- Finalmente otro multiplexor selecciona el circuito (S&H) que procede y se realiza la conversión a digital mediante un conversor de tipo SAR (*Successive Approximation Registers*) que mediante aproximaciones sucesivas, ajusta un valor digital a la lectura

analógica que hay en el condensador del S&H, guardándolo en el buffer correspondiente como último paso. Esta secuencia puede verse mejor en la Figura 3-10..

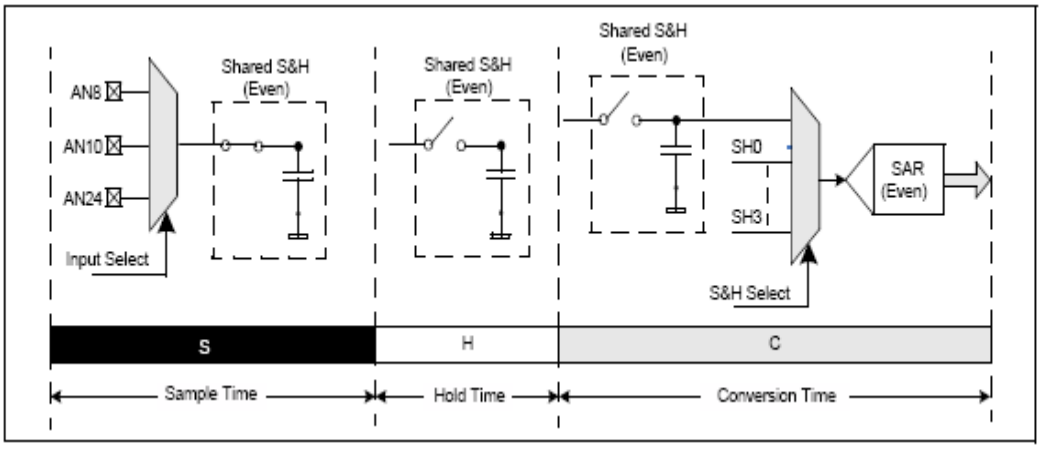


Figura 3-10 Secuencia de muestreo y conversión, imagen obtenida de [4]

3.1.5 El gestor de interrupciones

Las interrupciones son alteraciones o saltos en el flujo normal del programa para acceder a una subrutina, ejecutarse y después volver a la misma posición en la que se estaba. Esto se usa para que el programa pueda atender o actuar antes posibles entradas externas o internas. Una vez que el programa ha iniciado todos los registros necesarios, es habitual que el programa quede en un bucle infinito a la espera de estas interrupciones para proceder a hacer cambios o actuar. Para ello el programador debe configurar los registros propios de las interrupciones para habilitar aquellas que interesan, ya que el dsPIC33F dispone de hasta 118 fuentes de interrupción distintas.

Cuando se produce un evento que dispara una de estas interrupciones, el *Program Counter* (PC), salta hacía una posición de un vector de memoria donde cada interrupción tiene su posición. Este vector se conoce como IVT (*interrupt vector table*) y se muestra a continuación en la Figura 3-11.

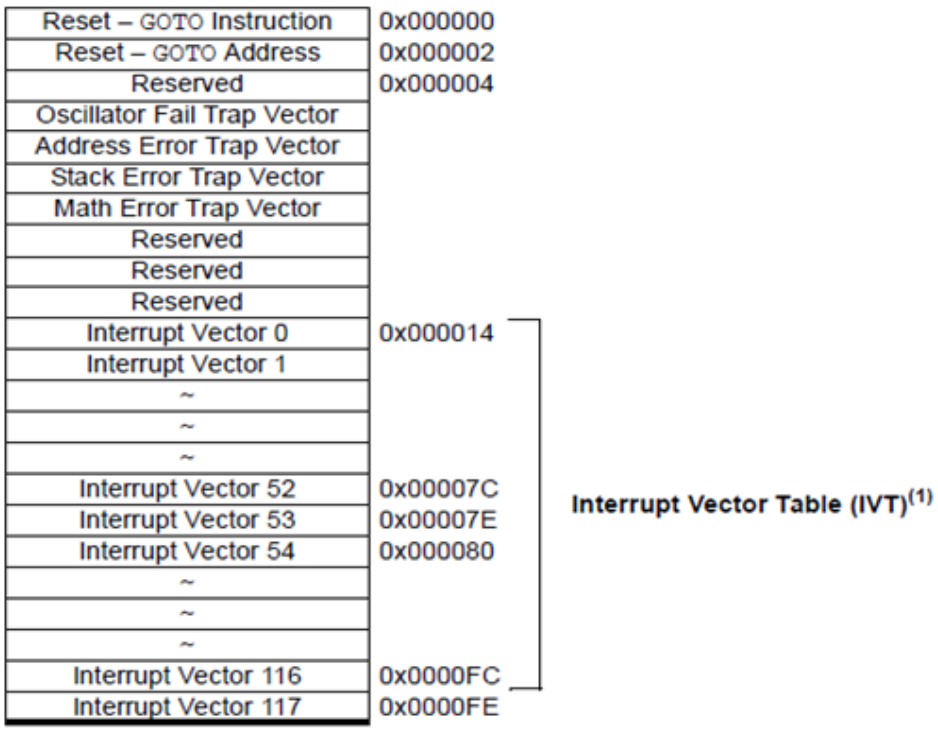


Figura 3-11 IVT Interrupt Vector Table, imagen obtenida de [4]

Dentro de la posición correspondiente del IVT, se tiene la dirección de memoria donde está el código que debe ejecutarse como consecuencia de la interrupción. Todas las interrupciones posibles tienen su dirección dentro del IVT, cuando una interrupción concreta se dé, el PC saltará a la propia posición de esa interrupción dentro del IVT y desde ahí, se irá a la dirección indicada en el vector. A continuación se ejecutará la subrutina y posteriormente el PC volverá a la función donde estaba antes de la interrupción para seguir con el flujo normal del programa, habiendo sido ejecutados los cambios programados en la interrupción.

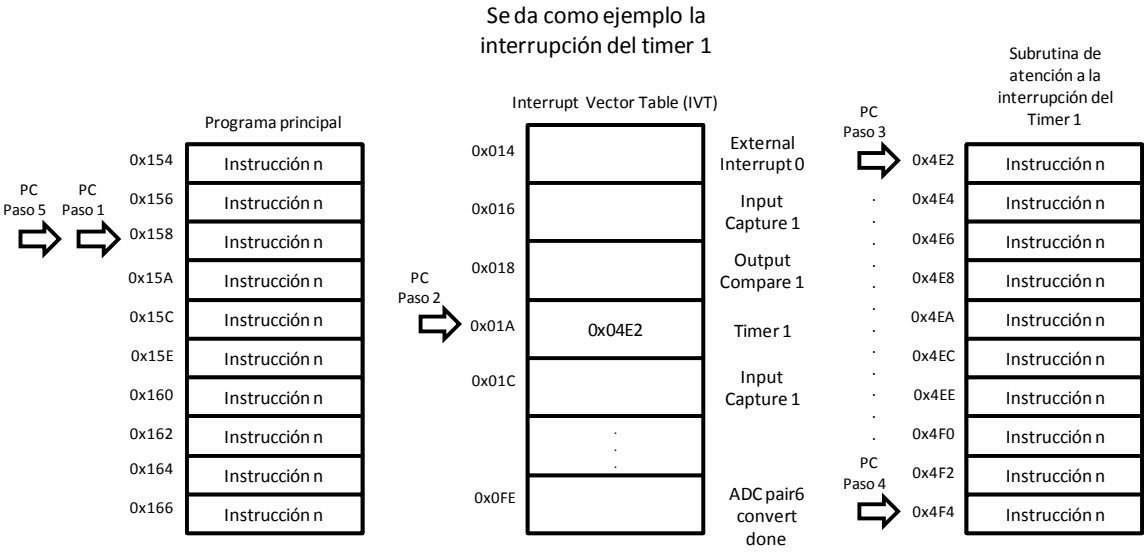


Figura 3-12 Pasos del PC (program counter) y flujo del programa ante interrupción

Para la configuración por parte del usuario de las interrupciones se actúa principalmente sobre los registros IFSx, IECx e IPCx.

Para la configuración por parte del usuario de las interrupciones se actúa principalmente sobre los registros IFSx, IECx e IPCx.

El IFSx, tiene en sus 16bits los distintos *flags* que indican que la interrupción se ha dado. Estos se ponen a 1 por hardware cuando se da la interrupción correspondiente y es el programa quien por software debe ponerlo a 0.

El IECx, es el que habilita la interrupción o interrupciones que el programador quiere tanto procedentes de periféricos (ADC) como externas (de botón) dentro de las 118 de que se disponen.

El IPCx, se usa para designar mayor o menor prioridad a las interrupciones. Cada una de estas tiene hasta 8 niveles prioridad configurables por el programador. En caso de darse dos interrupciones de la misma prioridad o en caso de no haberse configurado los nivel de estas, tiene más prioridad aquella que esta antes en el IVT. (La más próxima a la dirección 0x0000)

3.2 Sincronización del ADC y el PWM.

Debido a la especial importancia de esta sincronización entre el ADC y el PWM, se ha decido hacer una explicación más profunda ya que su configuración afecta en gran medida el funcionamiento del sistema completo.

Como ya se ha comentado anteriormente, las aplicaciones de electrónica de potencia con lazos de control de alta frecuencia requieren del funcionamiento sincronizado del módulo ADC y el PWM. De una forma simplificada se puede decir que el ADC lee el estado actual del convertidor y el PWM actúa sobre este para corregir defectos o mantener el régimen permanente. Por lo tanto cuanto menor sea el retardo entre la entrada al control (lectura ADC) y la actuación de este, (modificación del PWM) mejor será el control, siendo más rápido y teniendo unas características en la fase (dominio frecuencial) más próximas al caso ideal (caso de no existir retardo entre el sensado y la actuación). El módulo PWM contiene varios modos de interconexión con el ADC que se detallarán más adelante. Pero como idea general debe quedar claro que hay tres factores necesarios que son el PWM, el ADC y la interrupción. En general el PWM fija en qué momento y el evento que provoca la interrupción, el ADC es el que por ese evento dispara su interrupción y el gestor de interrupciones es el que interrumpe el flujo del programa para saltar a la interrupción solicitada por el ADC.

3.2.1 Registros de especial interés y su configuración

Para seleccionar una de las 16 fuentes que provocan la lectura, conversión e interrupción de ADC, se deben configurar los 4 bits de TRGSRCx que están en el registro de 16 bits ADCPCx. Mediante estos 4 bits podemos seleccionar la fuente que reclama la conversión. Este disparo se puede hacer individual por software para un ADC, global por software para todos los ADC, por PWM *SpecialEvent*, por Timer1 ó 2 y por *PWM generator* primario, secundario y por corriente límite. Hay que resaltar que se está trabajando con fuentes conmutadas y por lo tanto hay transitorios a la frecuencia de conmutación que provocan cambios más o menos bruscos en las variables físicas a controlar. Por lo tanto no es sólo importante la sincronización, sino también el instante en el que se hace la medida. Como ejemplo ilustrativo se muestra a continuación un elevador con un sensor de corriente colocado

a la fuente de un MOSFET de potencia, que supone una solución de bajo coste por no necesitar amplificación diferencial.

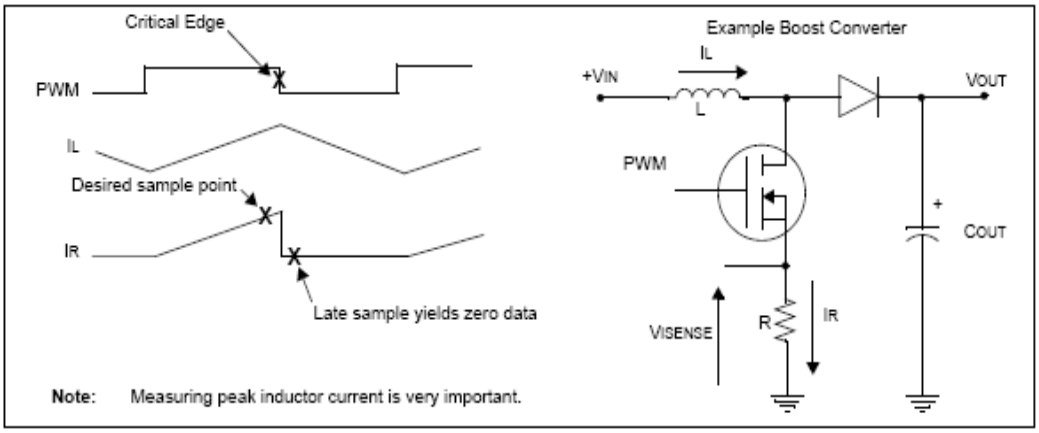


Figura 3-13 Ejemplo de lectura de ADC mal sincronizada con el PWM, imagen obtenida de [4]

Si se quiere controlar por corriente de pico, un retraso descontrolado provocaría una lectura de 0x0000 al intentar medir el pico de corriente. Con una buena configuración de esta sincronización ADC-PWM, esta situación puede ser resuelta.

Mediante las sincronizaciones por PWM primario, secundario y *specialevent*, se consigue que la solicitud de conversión no sea por software, lo que garantiza que la conversión y la interrupción se realicen en el instante necesario, ya que esta señal de disparo es por hardware, que no depende de retrasos causados por software. Por lo tanto para seleccionar la fuente del disparo se usarán los bits TRGSRCx del registro ADCPCx y para habilitar que se produzca la interrupción con el consiguiente salto en el flujo del programa, se debe configurar el bit IRQENx del registro ADCPCx del ADC. Para fijar un retardo conocido por hardware respecto al PWM usamos TRIGx para el primario o STRIGx para el secundario, que son registros del PWM.

Es posible también hacer que las interrupciones de ADC no se den con todos los flancos del PWM, es decir que se den a menos frecuencia que el PWM. Para ello se puede configurar este disparo cada 1, 2, 3 y hasta 16 flancos del PWM, así como esperar entre 0 y 63 ciclos del PWM antes de realizar la primera interrupción. Para esta espera se usan los bits TRGSTRT del registro TRGCONx y para la frecuencia de las interrupciones el TRGDIV del mismo TRGCONx del PWM. A modo de ejemplo se muestran dos casos de estas configuraciones. En el primer caso el disparo es cada PWM y arranca desde el primero donde el TRIGx es el retardo configurable comentado anteriormente.

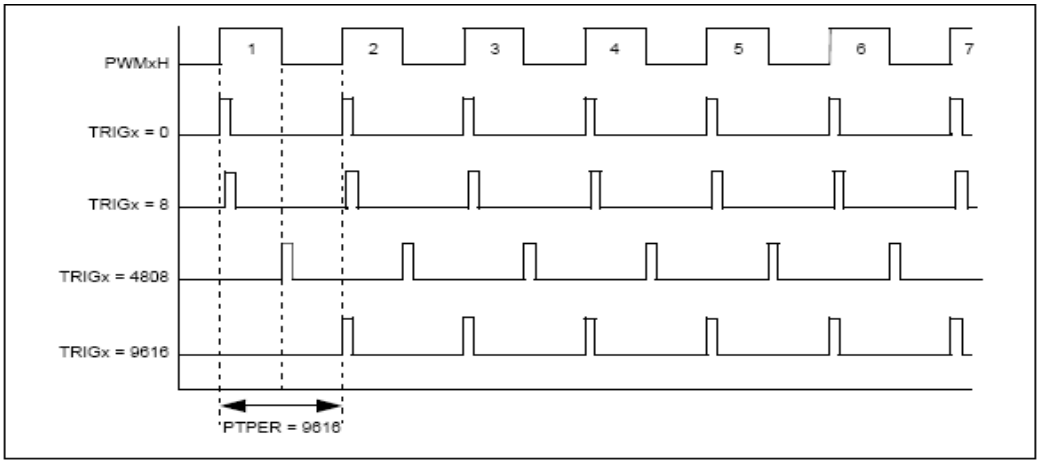


Figura 3-14 Disparo cada evento de PWM, imagen obtenida de [4]

En el segundo caso la interrupción se produce cada dos ciclos del PWM y empezando desde el segundo ciclo.

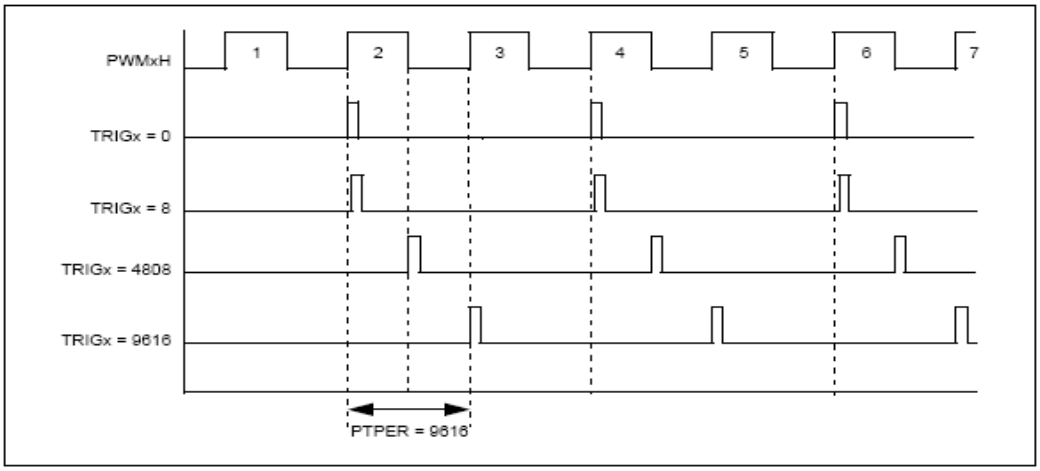


Figura 3-15 Disparo cada dos eventos de PWM, imagen obtenida de [4]

3.2.2 La interrupción del ADC

La conversión del ADC lleva asociada una interrupción si así lo quiere el programador. Esta función es de gran utilidad para conseguir que una subrutina sea ejecutada siempre que se dé un evento, que en el caso de convertidores conmutados el evento es el flanco de subida del PWM. Desde este salto en el flujo normal del programa, se puede ejecutar el algoritmo de compensación que realiza el cálculo de la nueva señal de control que se modulara con el PWM. Para ello se mete el nuevo valor de control en PDC, que es el registro de 16 bits que define la anchura del PWM dentro de su frecuencia de conmutación.

Esta flexibilidad en el instante en que se da la interrupción y su utilidad, permite ejecutar el algoritmo de control en el instante que más interesa al programador. Estas facultades permiten que el control digital mediante DSP, sea apropiado para el control de múltiples convertidores con un solo DSP. En la Figura 3-16 se muestra el ordenamiento de los algoritmos de control para dos convertidores controlados por el mismo DSP.

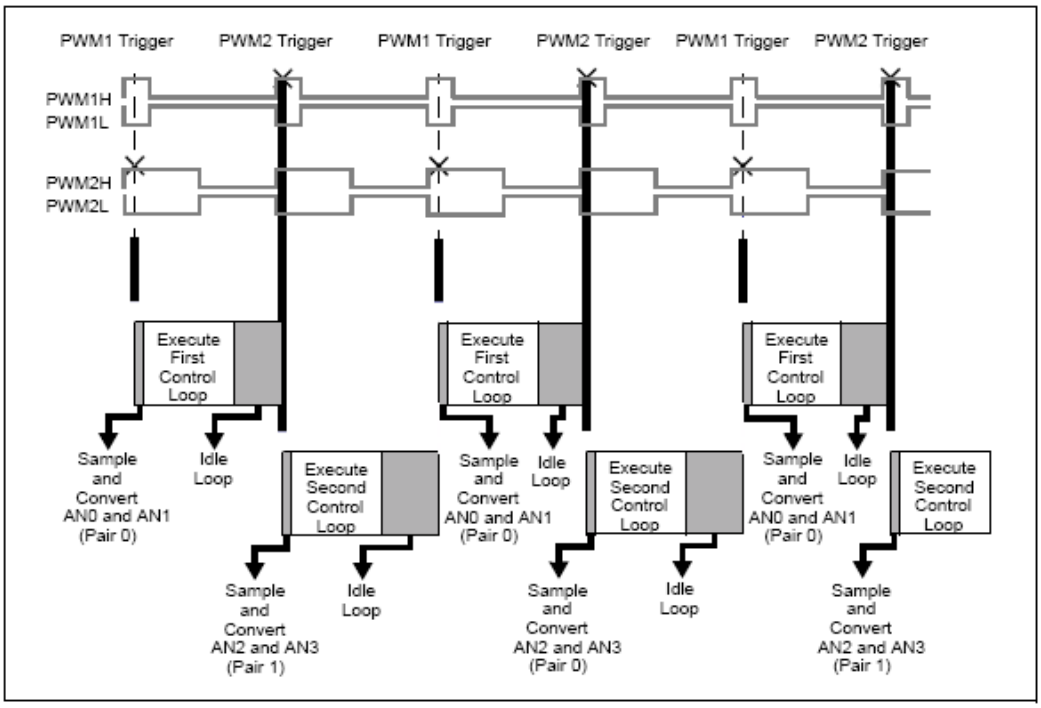


Figura 3-16 Control de 2 convertidores con un solo DSP, imagen obtenida de [4]

En este ejemplo vemos que no se producen interrupciones en todos los eventos del PWM. Tanto el PWM1 como el 2, provocan la interrupción cada 2 ciclos y para no solaparse, la interrupción del PWM2 está retrasada al segundo ciclo (en caso de solaparse se ejecutaría primero el de mayor prioridad y luego el de menos). Es decir, primero salta la interrupción de ADC provocada por el PWM1, luego en el siguiente evento la del PWM2 y así continuamente se van alternando para que se calculen los algoritmos de control en cada periodo, dando tiempo suficiente para ejecutarse todo desde un único DSP. Con ello y como se ve en el ejemplo se podrían controlar dos convertidores.

3.3 Resumen del dsPIC para el control de convertidores conmutados

En este apartado se han descrito las partes más importantes del dsPIC empleado en este proyecto con los periféricos que han sido usados para ello. Las características más importantes de este dsPIC para su uso como regulador digital de convertidores conmutados son las siguientes:

-Una CPU con hardware específico para realizar operaciones de tipo DSP, que permiten en un ciclo de máquina hacer operaciones como A x B = C. Estas han sido muy usadas en el algoritmo de control, por lo que es importante tener un módulo hardware que pueda hacer estos cálculos rápidamente.

-Un conversor analógico-digital, ya que en el lazo cerrado se trata de mantener una variable bajo control, por lo que es necesario conocer su magnitud y digitalizarla para poder en función de su valor, actuar sobre el sistema para enviar el valor real de la variable a controlar a su valor de referencia.

-Un generador PWM, ya que es este el que permite actuar sobre los semiconductores del convertidor conmutado para poder mantener bajo control la variable física.

-Un gestor de interrupciones que permita sincronizar PWM y ADC. De nuevo esto es importante ya que así las lecturas de las variables a controlar, se pueden hacer siempre en el mismo instante dentro del ciclo que fija el PWM.

Al disponer de estos módulos periféricos, se pueden estar realizando procesos de lectura y generación de pulsos PWM, independientemente de las instrucciones que se están ejecutando por el microprocesador. Esto resulta muy útil para el control de fuentes conmutadas, mientras se calcula un nuevo valor de control, se están realizando tanto lecturas de ADC como pulsos de PWM para mantener la variable a controlar siguiendo la referencia.

Es muy importante sincronizar estos procesos, ya que así las lecturas de ADC se hacen siempre en el mismo instante dentro del ciclo de conmutación del convertidor de potencia. De este modo se tienen valores de la variable a controlar que permiten realimentar el lazo y poder hacer cálculos de nuevos valores de control. Si tras la lectura se ejecuta el algoritmo de cálculo inmediatamente para obtener una nueva señal de control, se tendrá que el retardo o diferencia entre lo medido y lo actual es mínimo, lo que favorece la fase del lazo de control, favoreciendo la estabilidad del sistema.

Para la configuración exacta bit a bit de estos periféricos, véase el Anexo 11.1.1, donde se ve la configuración de cada registro comentando su función concreta.

4 Descripción de las herramientas y demoboards usadas en el proyecto

Para este proyecto se han usado las siguientes herramientas y DemoBoards del fabricante Microchip:

- **Buck/Boost Converter PICtail Plus Daughter Board** (Part Number: AC164133). Está compuesta por dos convertidores reductores y un elevador, por lo que en el ámbito de este proyecto no se ha dimensionado ninguno de los componentes de la placa ni se ha añadido nada a la misma. Estos convertidores, serán la planta del sistema a controlar por el dsPIC.

-**16-Bit 28-Pin Starter Board** (Part Number: DM300027). Es la placa donde está el dsPIC, tiene un conector de 28 pines especialmente diseñado para encajar en la placa Buck/Boost y un conector RJ-45 para poder programar y depurar el código del programa.

-**MPLAB ICD 2 debugger/programador**. Es la herramienta que se ha usado para programar los dsPIC y depurar los programas. Este dispositivo tiene a su salida un cable con conector RJ-45 para conectar con el starter board y por otro lado tiene una entrada que viene del puerto USB de un PC con MPLAB IDE.

-**MPLAD IDE v8.53**. Es el software usado para programar, desarrollar código y gestionar la comunicación a través del ICD2. Es el software gratuito que ofrece Microchip para el desarrollo de sus aplicaciones.

-**dsPIC33FJ16GS502**. Es el dsPIC utilizado en este proyecto por su especial orientación al control de convertidores de potencia y su compatibilidad con la placa 16-Bit 28-Pin Starter Board.

4.1 Buck/Boost Converter PICtail Plus Daughter Board

Esta es la placa que se ha usado como la planta a controlar en los lazos de control. De ella se han usado tanto los dos reductores síncronos como el elevador. Se ha optado por esta placa comercial y no por un prototipo propio por el interés en manejar componentes comerciales fáciles de conseguir, permitiendo así que todo el trabajo realizado en este proyecto pueda repetirse y ejecutarse usando estos mismos componentes comerciales sin tener que realizar un gran esfuerzo de diseño de prototipos. Esta placa tiene dos conectores principales, de los cuales en este proyecto solo se ha usado el de 28 pines que es el que permite conectar esta placa de convertidores, con la *StarterBoard* que se usará para el control de los mismos. Este conector principal tiene las siguientes conexiones dentro de la placa Buck/Boost (Tabla 4-1).

Pin Number Primary	Assignment	Primary Use
1	NC	Not Connected
2	AN0	Analog Input 0 (Buck Converter 1 current)
3	AN1	Analog Input 1 (Buck Converter 1 voltage)
4	AN2	Analog Input 2 (Buck Converter 2 current)
5	AN3	Analog Input 3 (Buck Converter 2 voltage)
6	AN4	Analog Input 4 (Input Voltage)
7	NC	Not Connected
8	VSS	Ground reference for logic and I/O pins
9	NC	Not Connected
10	NC	Not Connected
11	NC	Not Connected
12	NC	Not Connected
13	VDD	Positive supply for logic and I/O pins
14	NC	Not Connected
15	RB15	PMBUSAUX2
16	RB5	PMBUSAUX1
17	SCL	SCL/TX
18	SDA	SCL/RX
19	VSS	Ground reference for logic and I/O pins
20	VDD	+3.3V_DIG
21	PWM3H	PWM Boost Converter
22	I/O	Load
23	PWM2H	PWM 2 High Output (Buck Converter 2)
24	PWM2L	PWM 2 Low Output (Buck Converter 2)
25	PWM1H	PWM 1 High Output (Buck Converter 1)
26	PWM1L	PWM 1 Low Output (Buck Converter 1)
27	NC	Not Connected
28	+9V	Input Voltage for two buck stages

Tabla 4-1 Listado de pines del conector principal de la placa Buck/Boost

El conector principal está pensado para insertarlo en una placa de control, que reciba las señales analógicas de tensión y corriente y para que esta, le envíe los pulsos de control para los semiconductores. En la placa, el sensado de tensión se hace a través de un divisor de tensión resistivo y el de corriente mediante un subcircuito de sensado. Del mismo modo, los pulsos PWM que genera la placa de control son de 3.3V y estos pasan por un driver que permite el disparo de los MOSFET de potencia. Para ver el esquema electrónico detallado de las placas véanse los anexos 11.2, siendo la Figura 4-1 siguiente el esquemático simplificado.

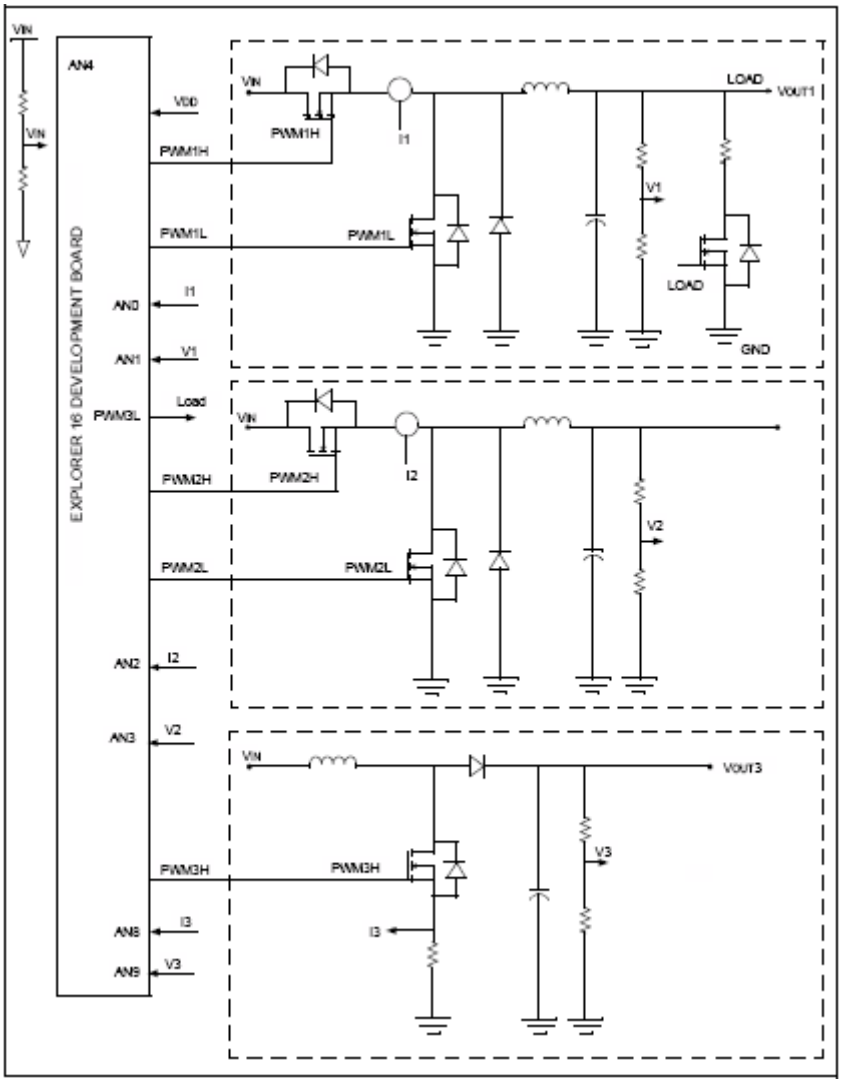


Figura 4-1 Esquema general de la placa Buck/Boost, imagen extraída de [5]

Es posible hacer que la alimentación tanto de la parte de potencia como de los drivers de la placa, se haga a través de la placa de control (pin28), pero para este proyecto se ha hecho de manera independiente a través una entrada auxiliar que se habilita simplemente abriendo un jumper. Se ha hecho así porque para las pruebas experimentales resulta útil poder variar la tensión de entrada y comprobar que el ciclo de trabajo varía para poder mantener fija la tensión a la salida del convertidor. De este modo, la entrada es común de los reductores y el elevador, pero es independiente de la entrada que alimenta la placa de control. A la salida de los convertidores se han puesto cargas resistivas de tal modo que los convertidores estén en modo de conducción continua (MCC), para las tensiones de salida que se van a manejar.

Con las placas Buck/Boost y starterboard de este proyecto, sólo es posible el control de dos de los convertidores, ya que sólo conectan cinco entradas analógicas. Una de ellas es para comprobar que la tensión de entrada esta dentro de unos límites y las otras cuatro, son para poder cerrar los lazos de control de dos de los convertidores con sensado de corriente y tensión. El reductor 1, siempre está cerrado y conectado a la placa de control, y el reductor 2 y el elevador comparten pines, por lo que se deben configurar cuatro jumpers para cerrar los lazos de tensión y corriente para el reductor 2 ó para el elevador, pero nunca los dos. De cara a

la parte experimental, estos jumpers han sido de gran utilidad ya que permitían introducir una perturbación AC en el lazo, sin tener que abrir el circuito por otro sitio, forzando la placa. Por lo tanto se consigue el propósito inicial de usar para todo, placas comerciales que se compran y están listas para hacer todo el proceso de diseño de reguladores sin tener que hacerles ninguna modificación de hardware.

Las características eléctricas a la salida de los convertidores reductores son una tensión de salida programable entre 0-5V con una corriente máxima de 3A. Y para el elevador la tensión máxima de salida es 20V con una corriente máxima de 0.75A. La placa dispone de muchos puntos de test para comprobar señales de realimentación, PWM, y tensiones de entrada y salida lo que la hace idónea para hacer test y pruebas ya que algunos han sido necesarios para medir la respuesta en frecuencia.

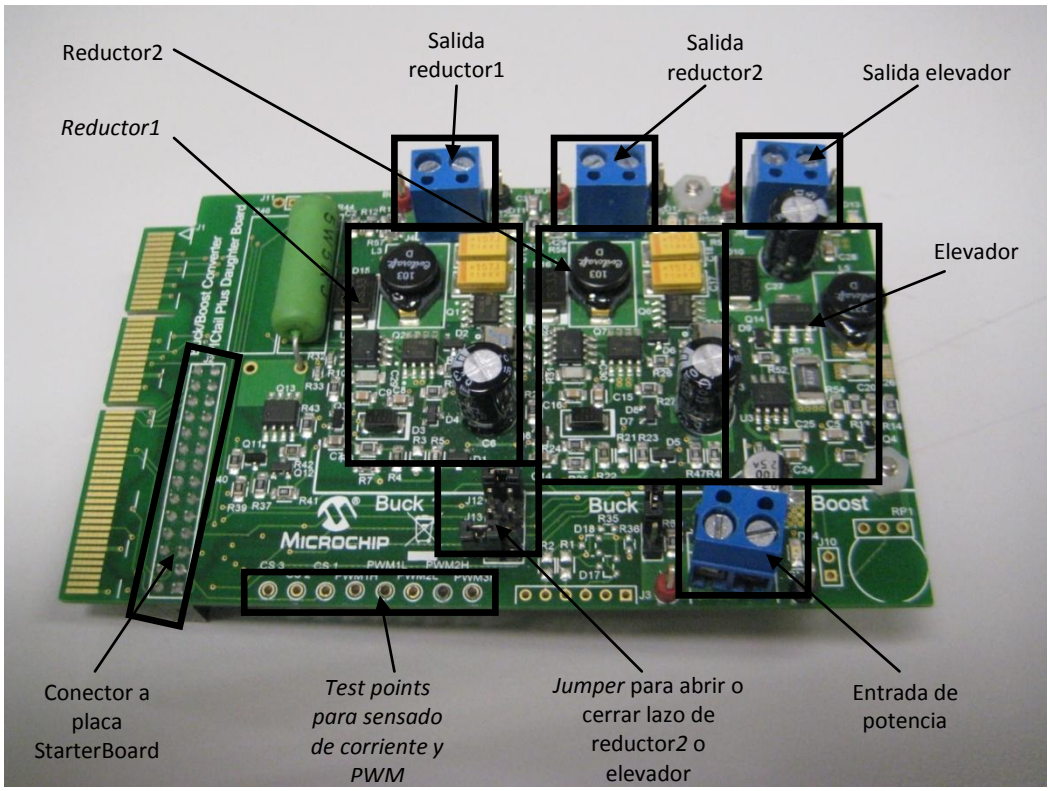


Figura 4-2 Placa Buck/Boost

4.2 16-Bit 28-Pin Starter Board

Sobre esta placa es donde se conecta el dsPIC que hace de control de la los convertidores de potencia. Esta placa no está exclusivamente pensada para esta aplicación por lo que el fabricante nos informa en la documentación que para usarla como controlador de la placa Buck/Boost es necesario retirar las resistencias R14 y R15. Esta es la única modificación que es necesaria hacer en todo el hardware del proyecto, para que todo pueda realizarse partiendo de estas placas. Esta manipulaciones una tarea sencilla y nada destructiva con un mínimo de cuidado. La placa dispone de dos botones, uno reset y otro configurable que ha sido usado para que el usuario pueda hacer cambios en la referencia para poder ver la respuesta temporal del lazo ante escalones en la referencia

El elemento más significativo de esta placa es el dsPIC33F, ya descrito en apartados anteriores.

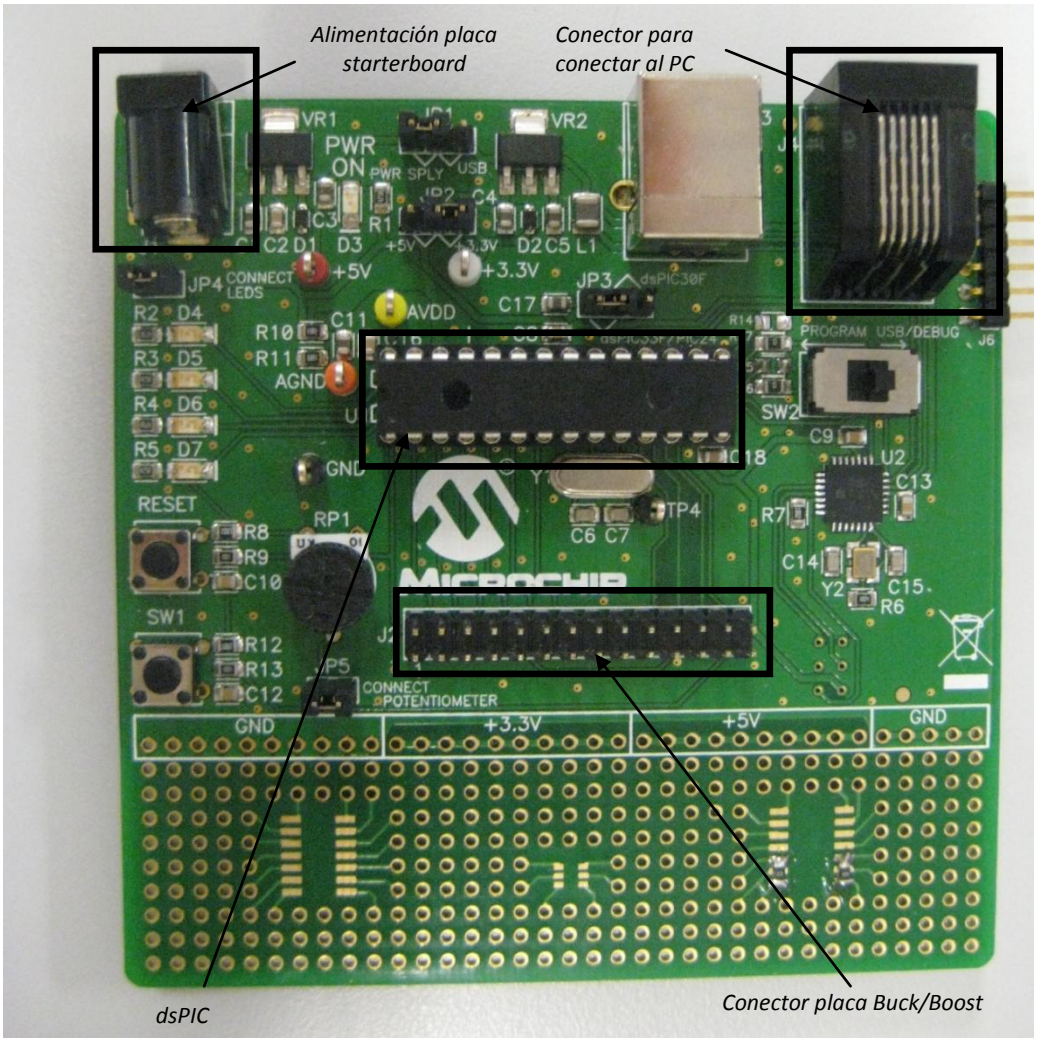


Figura 4-3 Placa de control

4.3 MPLAB ICD 2

Este es el dispositivo que se ha usado para la programación y depurado de los programas con la ayuda del MPLAD IDE. Este tiene sólo una entrada y una salida. La entrada es un cable USB, que se conecta a un PC. A su salida tiene un cable RJ-45, para conectar el ICD2, con la placa starterboard y de ahí con el dsPIC, para poder programar y depurar programas. El ICD2 sólo necesita para funcionar que se instalen los drivers correspondientes en el PC, sin tener que hacer ninguna operación más para comunicar PC y dsPIC. El la Figura 4-4, puede verse el ICD2 y sus cables de conexión.

Figura 4-4 Depurador de Microchip, MPLAB ICD2

4.4 MPLAB IDE v8.53

El MPLAB IDE es el software gratuito que ofrece Microchip para configurar, programar y depurar sus microcontroladores. En la *Figura* 4-5, se puede ver la vista principal del software.

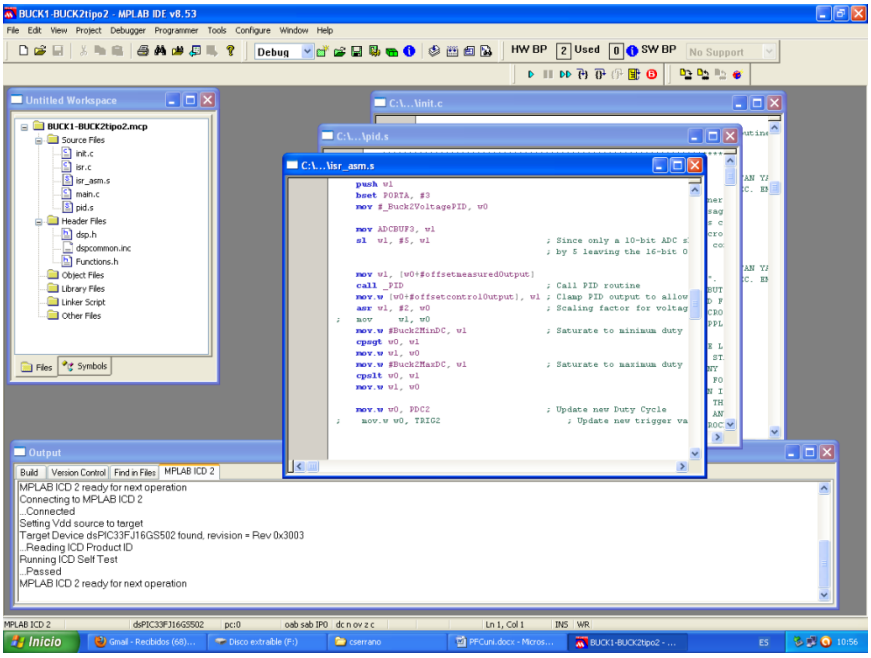


Figura 4-5 Software Microchip, MPLAB IDE

Este programa es el que ha sido utilizado para editar los programas, depurarlos y también programar el dsPIC. A continuación se hará una descripción de las funciones usadas en el ámbito de este proyecto.

Lo primero que se debe hacer para que se pueda realizar el depurado y la programación es sincronizar el MPLAB IDE y el dsPIC. Para ello se selecciona el dsPIC que se va a usar, en la ventana "*Select Device*" como se ve en la Figura 4-6. Después se sincronizan el reloj del dsPIC y del ICD2 además de otros bits de configuración como se ve en la Figura 4-7. Estos dos menús se encuentran en el menú desplegable "configure".

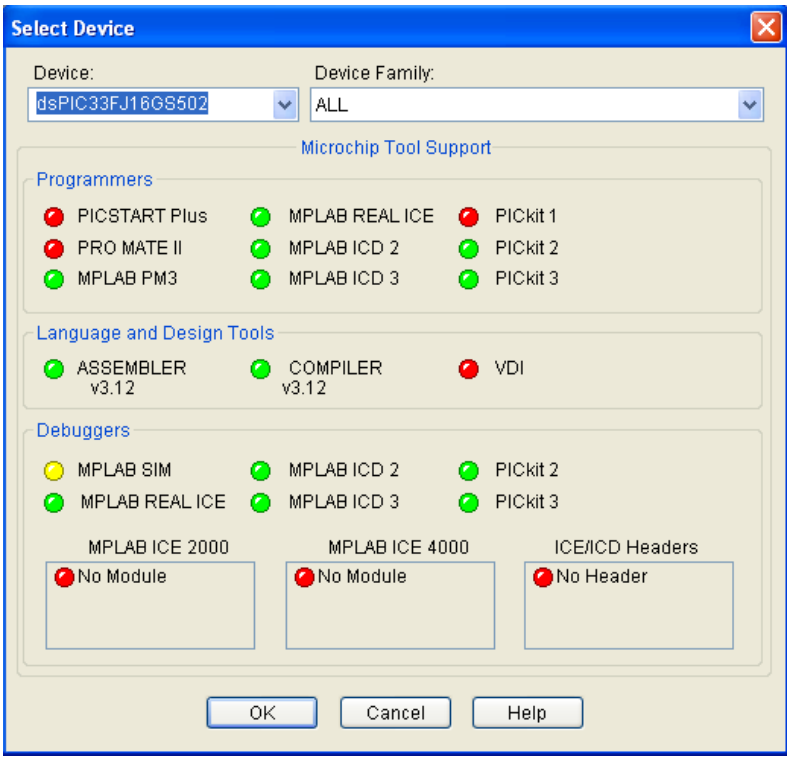


Figura 4-6 Ventana de selección del dsPIC

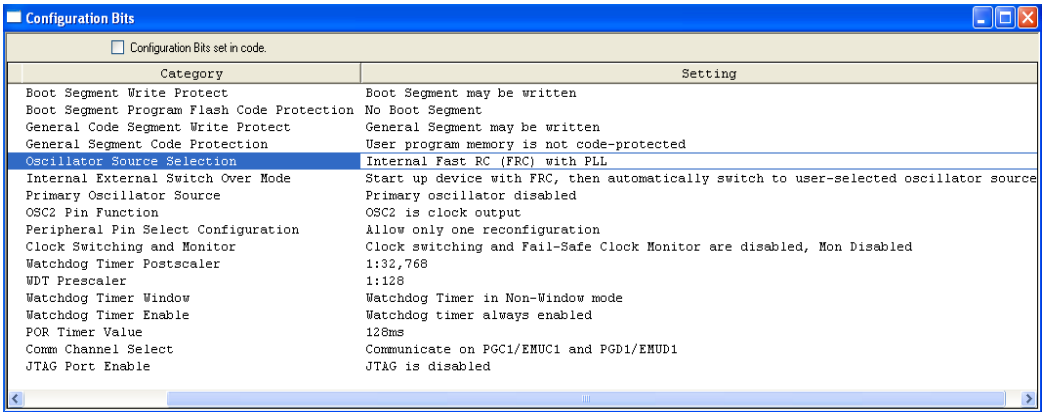


Figura 4-7 Ventana de configuración de Bits

En el menú "*Configuration Bits*", es importante decir que todos los bits están por defecto configurados para que se pueda programar y depurar perfectamente todo, excepto el que se ve marcado en la Figura 4-7, que debe estar con la opción "*InternalFast RC* (FRC) *with PLL*".

Una vez se ha hecho esto y está seleccionado o abierto un proyecto, se selecciona en el menú desplegable, el modo *debugger→select tool→*MPLAB ICD2 para hacer el depurado o *programmer→select programmer→*MPLAB ICD2, para programar definitivamente el dsPIC. A continuación se describirán las funciones que han sido usadas en este programa para depurar y programar.

4.4.1 El depurador

La ventana mostrada en la Figura 4-8, es la que corresponde al MPLAB en modo de depurador. Este modo ha sido muy usado en este proyecto para poder seguir el código programado y su funcionamiento en cada instrucción. Para realizar el seguimiento detallado del código programado se han usado especialmente las siguientes ventanas del depurador de MPLAB IDE.

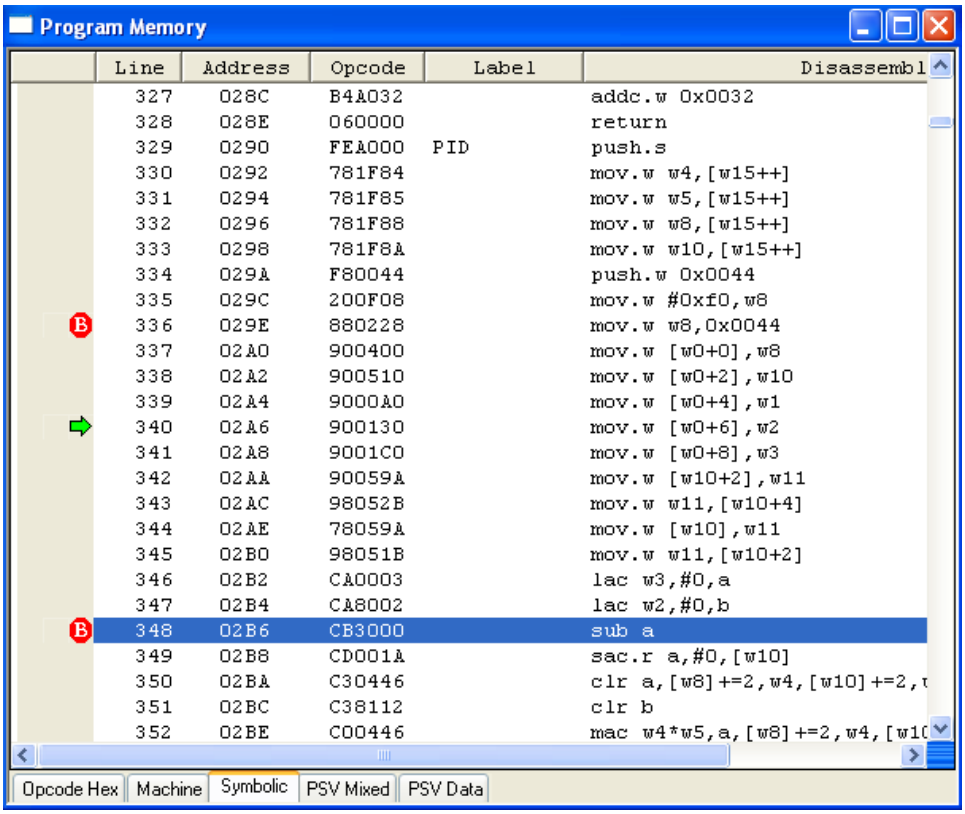


Figura 4-8 Memoria de programa, para seguir el flujo del mismo

La primera de las ventanas del depurador utilizadas es la que se muestra en la Figura 4-8.En esta ventana se ve el programa real implementado en el dsPIC, con todas las instrucciones de ensamblador con su equivalente en código máquina. En esta ventana, incluso las partes de código escritas en C, se ven ya en código máquina. Esta herramienta, permite ejecutar las instrucciones paso a paso o en modo normal, donde el programador puede insertar puntos donde parar la ejecución (*breakpoint*) y de nuevo ejecutar paso a paso o normalmente para poder seguir detalladamente el proceso secuencial del programa.

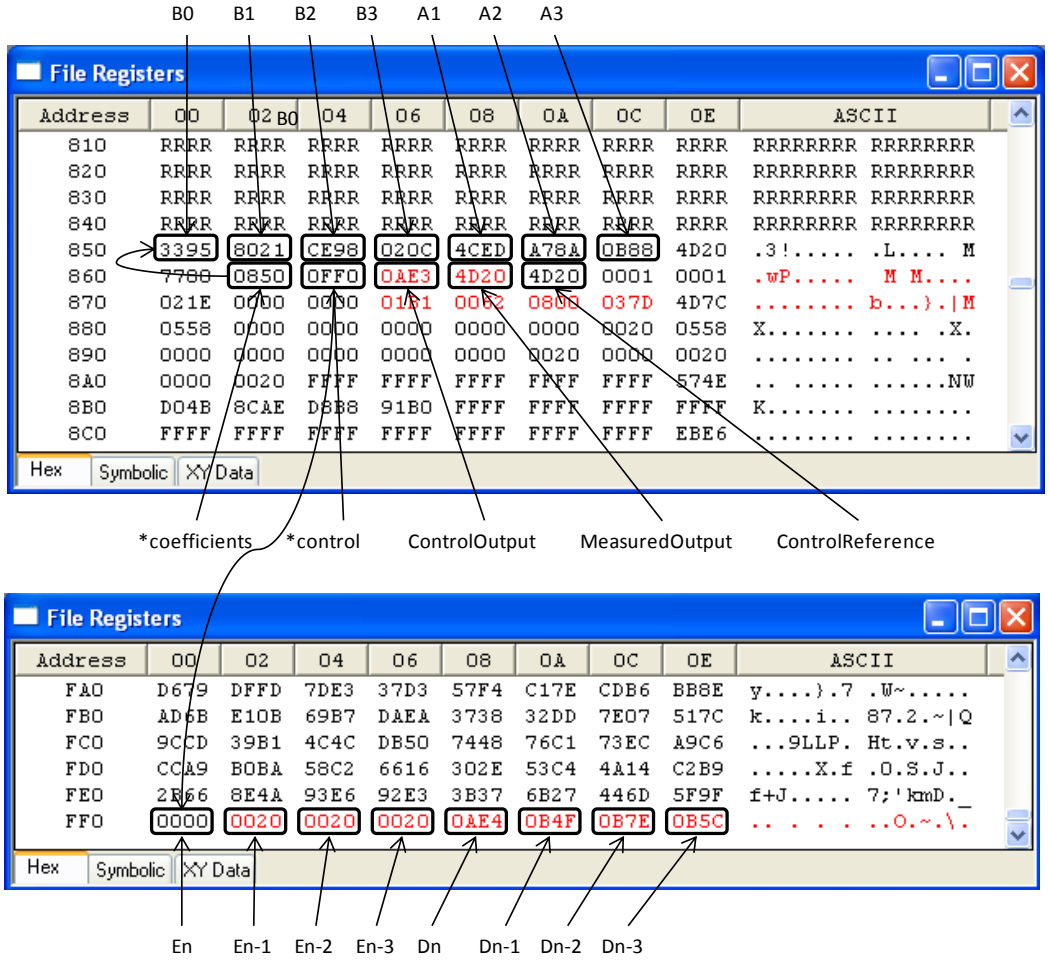


Figura 4-9 Memoria de datos y sus significados en el programa

Otra ventana muy usada en el depurado de programas es la "*File Registers*" que se ve en la Figura 4-9 y que muestra el contenido de la memoria. En ella se puede ver perfectamente cómo es una variable tPID, que se explicará detalladamente en el apartado 6.2.2.3 y que está compuesta por los punteros **coefficients*, **control*, que apuntan a donde están los coeficientes y los valores de error y ciclo de trabajo, más *ControlOutput*, *MesuredOutput* y *ControlReference*, que son variables en memoria de datos usadas para hacer el cálculo de la señal de control. De esta variable se hablará más detalladamente en el punto 6.2.2.3, pero aquí se puede ver un caso real de la misma. Esta ventana de "*File Registers*" ha sido de gran utilidad para ver como se hacía el cálculo del control y como las operaciones MAC multiplicaban los valores de coeficientes, error y ciclo de trabajo correctamente en el orden que el programador esperaba.

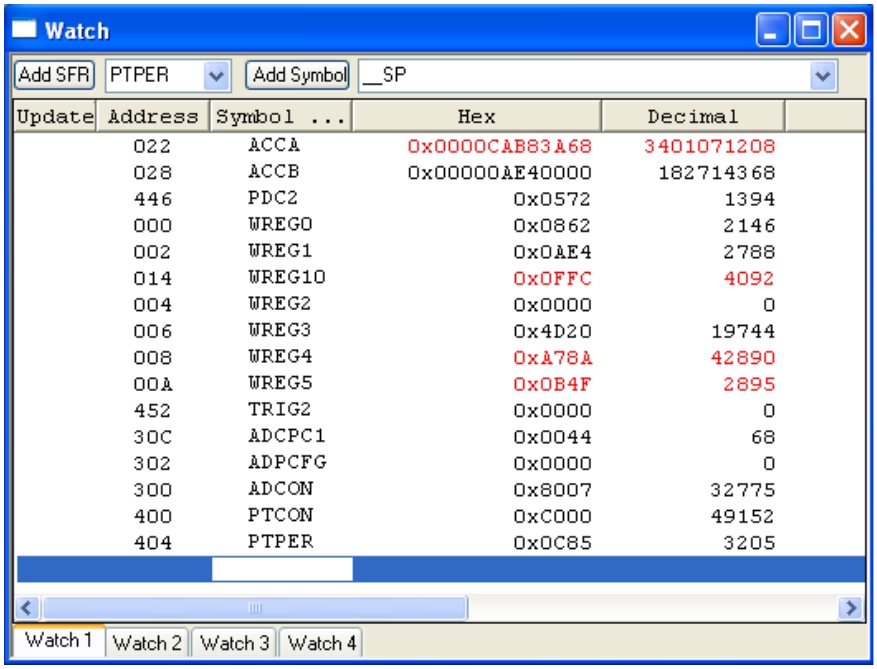


Figura 4-10 Ventana "Watch", seguimiento de registros especiales

Por último otra ventana que ha sido muy usada en el depurado de los programas es la "*Watch*", que se ve en la Figura 4-10.En esta ventana se pueden ver los registros que interesen al programador incluyéndose los registros de trabajo, pudiendo ser buscados por su nombre en lugar de por su dirección.

Para terminar con la descripción del *debugger* o depurador, hay que decir que estas tres funcionalidades anteriores han sido muy usadas para poder seguir, entender y corregir los programas para conseguir que el sistema funcionase. Los tres eran usados en paralelo, ya que por separado no aportan suficiente información. Sin embargo usados conjuntamente permiten hacer un depurado detallado del programa. Por lo tanto, en el proceso final de desarrollo de programas, la vista más habitual del MPLAB IDE es la vista en la Figura 4-11, donde se ven las ventanas "*ProgramMemory*", "*File Registers*", "*Watch*" y la ventana de uno de los ficheros de código, en el ejemplo de la figura la PID.s. Aparte de lo comentado anteriormente, normalmente se han usado simplemente los botones de la barra de herramientas como "*Buildall*" para compilar todo, "*Program target device*" para programar, y después los propios del depurador. Estos son "*Run*", para que corra el programa con normalidad, "*Halt*" para pararlo y "*Reset*", para resetear el programa es decir, para llevar el ProgramCounter a la posición de memoria de programa 0000. Además se han usado los que permiten correr el programa paso a paso que son, el "*stepinto*" y "*stepover*", donde el primero ejecuta la instrucción y si es una función entra para hacerla paso a paso, y el segundo, que ejecuta la instrucción y si es una función la ejecuta entera sin mostrar el proceso paso a paso interno de la función.

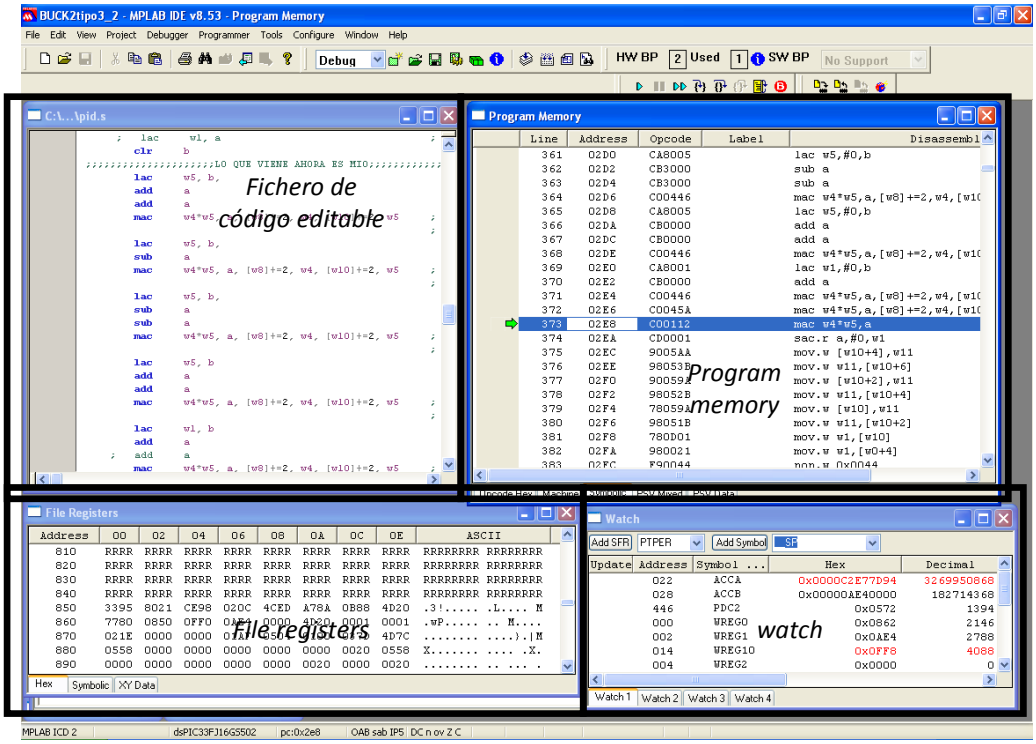


Figura 4-11 Vista general del MPLAB IDE durante el depurado de programas para este proyecto

Finalmente una vez era comprobado que el programa funcionaba correctamente, se pasaba al modo programar.

4.4.2 El programador

En modo programador están todas las ventanas descritas anteriormente disponibles, pero en realidad no resultan útiles ya que en este modo se programa el dsPIC para que corra independientemente del PC, con lo que no se puede mostrar en funcionamiento la información descrita en el punto anterior. En este caso la ventana principal se ve en Figura 4-12. Esta ventana por lo tanto se diferencia de la del depurador muy poco. Sin embargo en esta los únicos botones usados han sido "*Buildall*", para compilar todo, "*Program target device*", para programar y finalmente, "*ReleasefromReset*", para hacer activo el programa ya en el dsPIC y poder desconectar las placas del depurador y por lo tanto del PC.

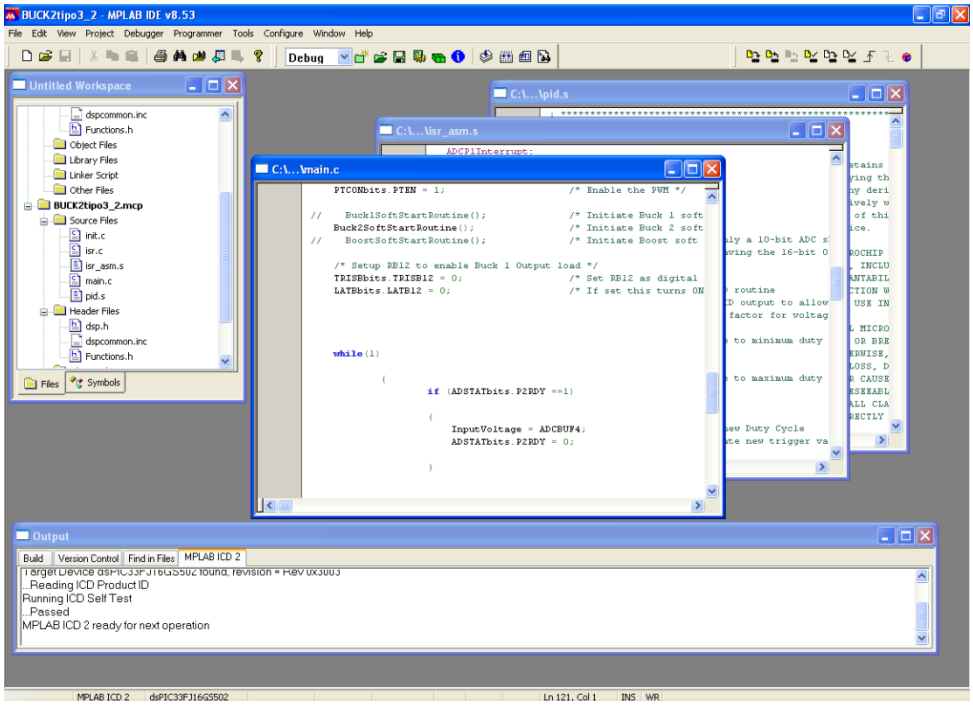


Figura 4-12 Vista general del MPLAB IDE durante el programado del dsPIC

4.5 El montaje de los convertidores con control en lazo cerrado

Una vez se han explicado todas las placas y herramientas usadas para que los convertidores funcionen en lazo cerrado, se van a comentar algunos detalles que son importantes para el proyecto. La idea de hacerlo todo con productos comerciales fácilmente accesibles, ha limitado algunos aspectos del prototipo. Por ejemplo, como se ha comentado anteriormente, sólo se han podido controlar dos convertidores a la vez, aunque este dsPIC puede controlar hasta 4, ya que tiene 4 pares de entradas analógicas, por lo que cada par puede leer tensión y corriente. También tiene hasta 4 generadores PWM con sus respectivas señales complementarias PWMxH y PWMxL. Otra limitación muy importante es que la conexión entre las señales de entrada y salida de la tarjeta de potencia Buck/Boost (señales PWM, señales analógica) y los pines del dsPIC está ya determinada por el rutado y los conectores de las dos tarjetas. El margen de maniobra para reconectar señales o sacar al exterior del dsPIC señales nuevas es muy pequeño. Por ejemplo, para poder usar el botón de la placa *starterboard* para cambiar la referencia del lazo de control, es necesario remapear el pin del dsPIC que está conectado físicamente al botón, (pin 14 del conector J2 de la placa *starterboard*) y asociarlo a una interrupción Algo similar pasa para poder ver la señal binaria que monitoriza el tiempo que tarda en ejecutarse un algoritmo de control, como se explicará más adelante. Como no hay otro punto del dsPIC que vaya a un punto de test donde poder ver la señal con el osciloscopio, se usó el PWM1L. Este se remapeó como salida digital, para poder verlo en el punto de test del PWM1L de la placa Buck/Boost. Lógicamente para ello se dejó inactivo el PWM1 y con el ello el reductor 1, haciendo la monitorización de dicha señal solo para el reductor 2 y el elevador. Esto resulta más una molestia que un problema, ya que conociendo el tiempo que tarda el reductor 2, se sabe lo que tarda el 1. Pero lo que debe quedar claro es que no se puede hacer todo a la vez porque con este montaje queda todo muy ajustado para manejar solamente 2 convertidores.

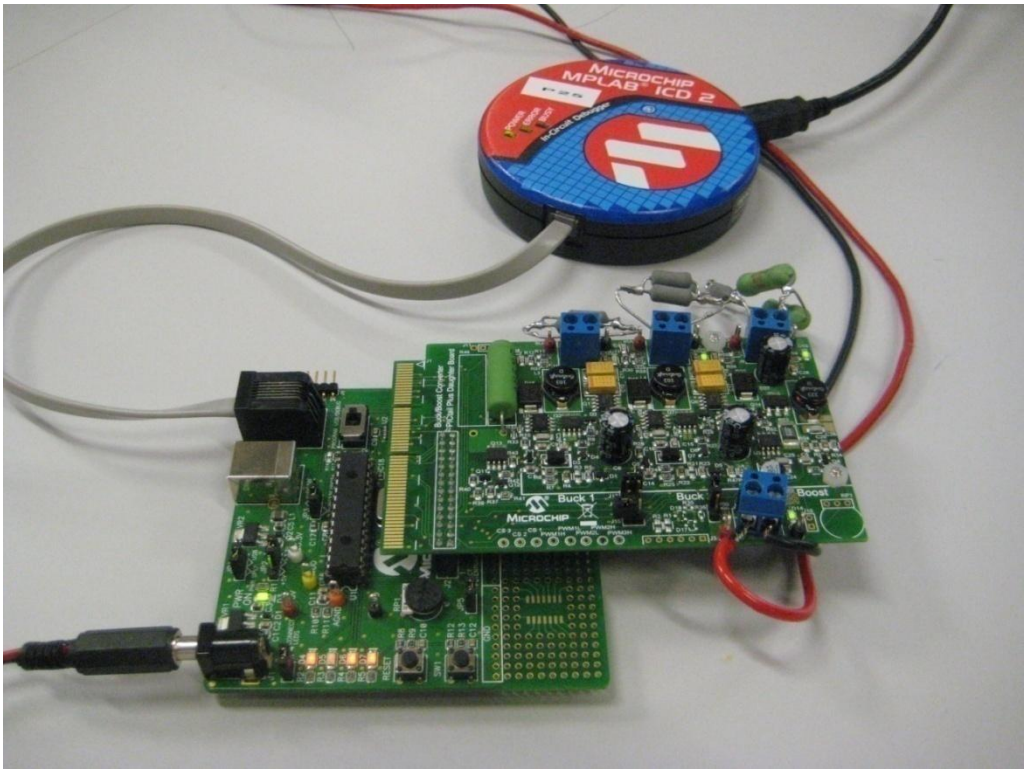


Figura 4-13 Montaje experimental con el depurador-programador

En las condiciones de montaje descritas debe señalarse lo siguiente:

- Solo pueden funcionar conjuntamente reductor 1 y 2 o reductor 1 y elevador
- Reductor 2 y elevador nunca pueden ir juntos porque comparten pin a la hora de cerrar el lazo desde la placa *Buck*/*Boost* a la *StarterBoard*
- Para las medidas experimentales de respuesta en frecuencia, sólo se usan reductor 2 ó elevador (nunca juntos), ya que sólo estos tienen jumper disponible para abrir el lazo e inyectar la perturbación.
- Para medir el tiempo que se tarda en calcular el ciclo de trabajo, es necesario usar un pin accesible de la placa *Buck*/*Boost*, por lo que es necesario usar una pata PWM que en este proyecto fue la PWM1L. Por lo tanto no es posible ver este fenómeno con el reductor 1 funcionando. Es decir, el reductor 1 debe ser deshabilitado para medir el tiempo que se tarda en hacer el cálculo de la señal de control.

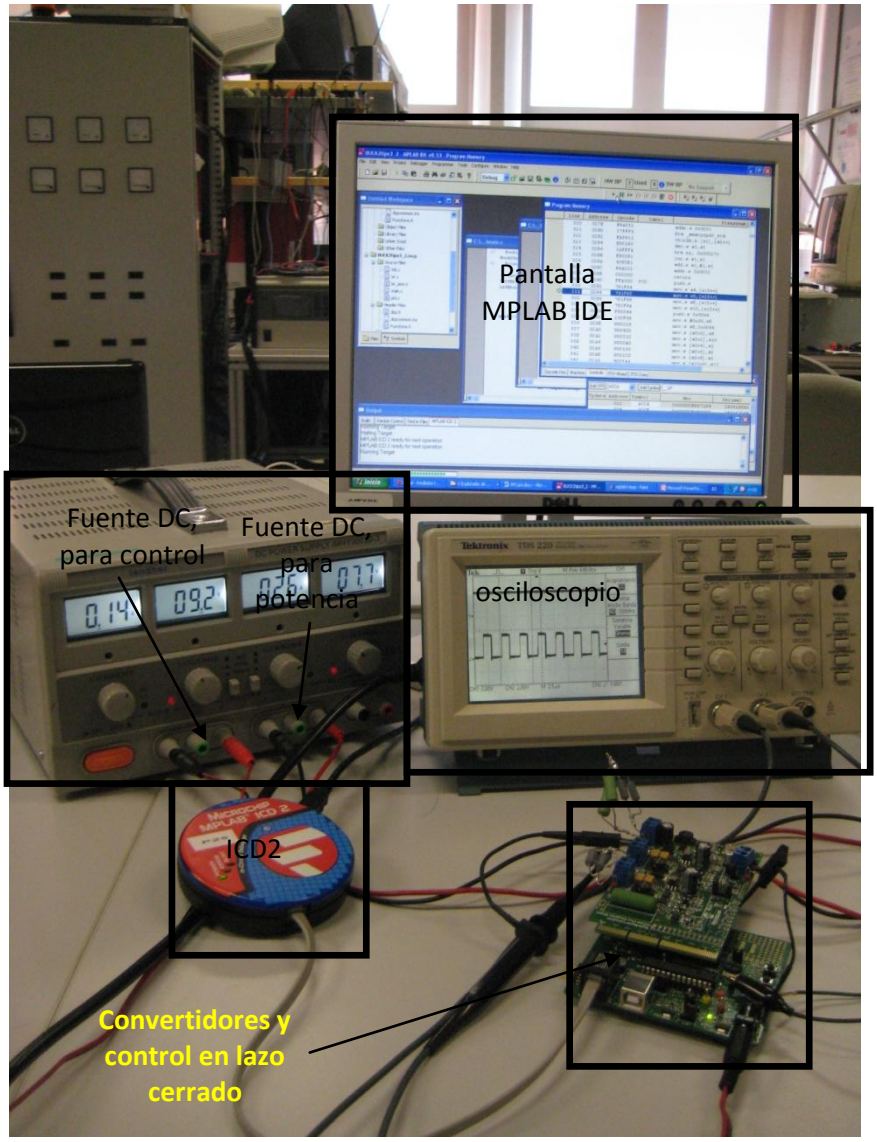


Figura 4-14 Montaje general para depurado y desarrollo de programas para las aplicaciones de potencia de este proyecto

Como imágenes ilustrativas se muestran la Figura 4-13 donde se ve el montaje compuesto por las placas *Buck*/*Boost* y la *starterboard* con el ICD2. De este montaje el ICD2 sólo es necesario usarlo si se está depurando el programa o en el momento de la programación, pero con el dsPIC programado, lógicamente no hace falta y el montaje queda simplemente con las placas. En la Figura 4-14, se ve el montaje general con el que se han desarrollado los programas del dsPIC. Consta del montaje anterior más el MPLAB IDE y un osciloscopio para poder ver si se realiza el control de la tensión de salida o no. En el ejemplo de la foto se ve que los pulsos del PWMxH, funcionan correctamente con lo que la tensión de salida está controlada.

5 Metodología de diseño de lazos de control digital basados en dsPIC

En este capítulo se va a explicar la metodología seguida para el diseño de los reguladores implementados en el dsPIC, para el control de los convertidores de la placa *Buck*/*Boost*.

Una de las motivaciones de este proyecto ha sido la validación experimental del programa de diseño de reguladores SmartCtrl, realizado en el GSEP y comercializado internacionalmente por *Powersim Technologies*, dentro del paquete PSIM. Por lo tanto se ha seguido una metodología de diseño compatible con el software utilizado.

Para el diseño sistemático de reguladores se han usado las siguientes herramientas: SmartCtrl de PSIM, MATLAB y el ya mencionado MPLAB IDE de Microchip. A continuación se describirán las herramientas y como han sido usadas en el proceso de diseño del regulador.

5.1 SmartCtrl

Como se comentó en el punto 2.3.3 sobre el diseño de controles digitales, es posible hacer el diseño directamente en discreto o hacerlo primero en continuo, discretizarlo y después adaptarlo a su implementación final. En este proyecto, se ha optado por la opción de hacer el diseño del regulador en tiempo continuo y después discretizar para así poder usar la herramienta SmartCtrl en la versión que se dispone en el laboratorio, que no incluye la parte de control digital. Por lo tanto se usará SmartCtrl para hacer el diseño del regulador en continuo. Esta herramienta permite diseñar de forma rápida y sencilla reguladores tipo3, tipo2, PI o polo simple, centrándose principalmente en su margen de fase y frecuencia de cruce del lazo abierto del sistema.

El proceso de diseño son SmartCtrl es el siguiente:

- Definición de la planta
- Definición del sensor
- Definición de los objetivos dinámicos (margen de fase y frecuencia de corte)
- Obtención del regulador y análisis del mismo.

5.1.1 La planta

SmartCtrl permite introducir la planta mediante topologías predefinidas (reductor, elevador, reductor elevador, Flyback, etc) o importar su respuesta en frecuencia en formato numérico, ya sea desde un barrido en frecuencia de un simulador (ej. *AC sweep* del mismo PSIM), o mediante la medida experimental de la planta mediante un analizador de ganancia y fase. En el caso de este proyecto se ha importado la planta desde los resultados de medidas experimentales con el analizador de ganancia y fase. Es decir, se ha medido la respuesta en frecuencia de la planta en el laboratorio y se han extraído los datos en formato numérico para introducirlos en SmartCtrl.

En el caso de este proyecto se tiene un lazo con control digital, que conlleva en retardo en la respuesta en frecuencia que hay que considerar en el cálculo del lazo. En la herramienta SmartCtrl que se ha usado, no se dispone de funciones para lazos digitales y por lo tanto la

introducción del retardo, que es propio de lo digital y no de lo analógico, no se puede introducir en el regulador, que es el responsable real de ese retardo. Por lo tanto para que en el cálculo del lazo de SmartCtrl se tenga en cuenta esta particularidad, ha sido necesario introducir este retardo en la respuesta en frecuencia de la planta que se importaba, añadiendo a los datos obtenidos en la medida el efecto del retardo.

Este retardo afecta a la fase y es mayor cuanto mayor es la frecuencia, ya que ese retardo, que es un tiempo fijo, es más significativo cuanto menor es el periodo, y por tanto mayor es la frecuencia. Esta variación en la fase afecta a la estabilidad del lazo por lo que es importante incluirlo en la planta que se importa a SmartCtrl, teniendo así que el cálculo de la respuesta en frecuencia de la ganancia de lazo abierto se ajustará más a al comportamiento real. En caso de no añadir este retardo, es fácil cometer el error de fijar un margen de fase sobre este sistema sin retardo que puede ser insuficiente en el caso real que sí tiene retardo, haciendo que el montaje experimental sea inestable cuando en SmartCtrl se tiene que debería ser estable. En las figuras siguientes se ve la planta importada a SmartCtrl en este proyecto, donde la ganancia está en la Figura 5-1 y la fase en la Figura 5-2. Así mismo, en la fase se puede ver la gran diferencia entre la fase en caso de añadir el retardo o no hacerlo.

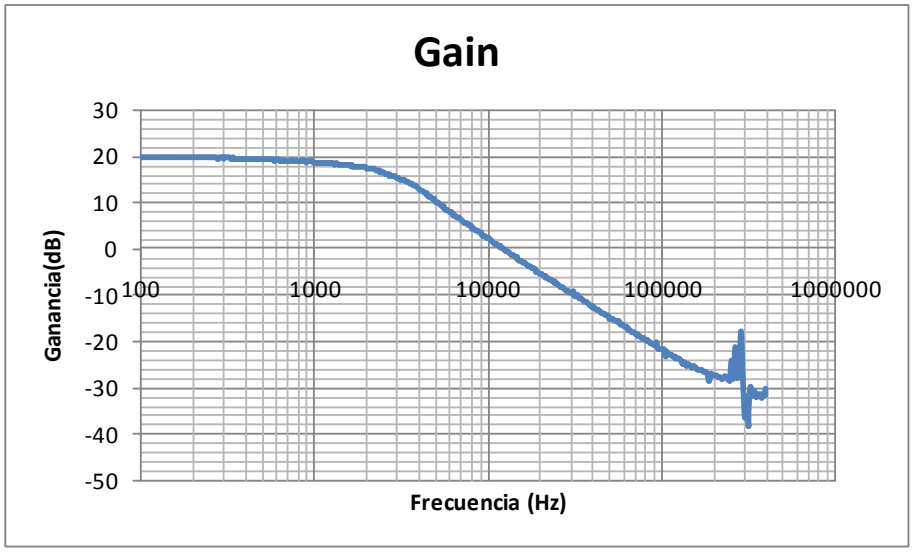


Figura 5-1 Ganancia de la planta importada a SmartCtrl

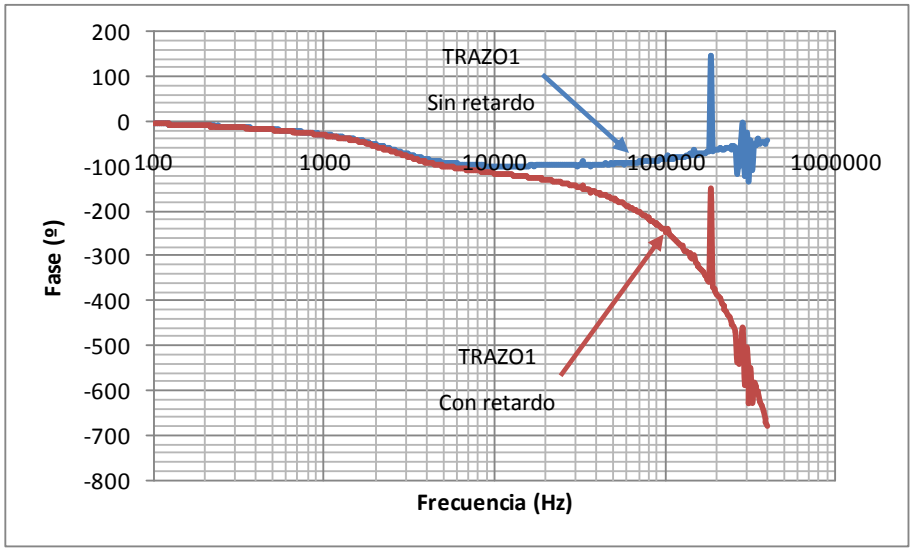


Figura 5-2 Fase de la planta importada a SmartCtrl y la misma sin retardo

Para este proyecto se ha usado como es lógico, la planta que incluye el retardo. Como puede verse a mayores frecuencias la fase cae bruscamente con lo que el margen de fase, si no se incluye la planta con retardo, puede quedar totalmente falseado respecto a lo que indica SmartCtrl. Como se ha comentado anteriormente, la fase cae con la frecuencia, ya que el retardo es siempre un tiempo constante, que depende del tiempo que tarda en actualizarse el PWM y que es más significativo cuanto mayor es la. Para introducir este retardo dependiente de la frecuencia en la fase de la planta se ha utilizado la siguiente expresión que implementada en Excel ofrece la fase con el retardo en función de la frecuencia f:

$$Retardo(f) = T_{retardo} \cdot f \cdot 360 \qquad \textit{Ec. 5-1}$$

Donde $T_{retardo}$ es:

$$T_{retardo} = 1.33 \times T_{con} \qquad \textit{Ec. 5-2}$$

El tiempo de retardo considerado es igual al tiempo que transcurre desde que el ADC muestre la señal controlada (tensión de salida en este caso) hasta que finaliza el pulso de control que se generó con dicha muestra, es decir, lo que tarda realmente en actualizarse el PWM. En este caso el retardo considerado es de 1.33*T_{con}, ya que los cálculos del nuevo valor de control se realizan a partir de un evento en el PWM y actualización de este nuevo valor una vez calculado, se realiza en el siguiente periodo de PWM. Como en régimen permanente se tiene que Vin=10 y Vout=3.3, el ciclo de trabajo del PWM es aproximadamente 0.33 en todo momento, pero con pequeñas variaciones en torno a este valor de d, como se ve en la en Figura 5-3.

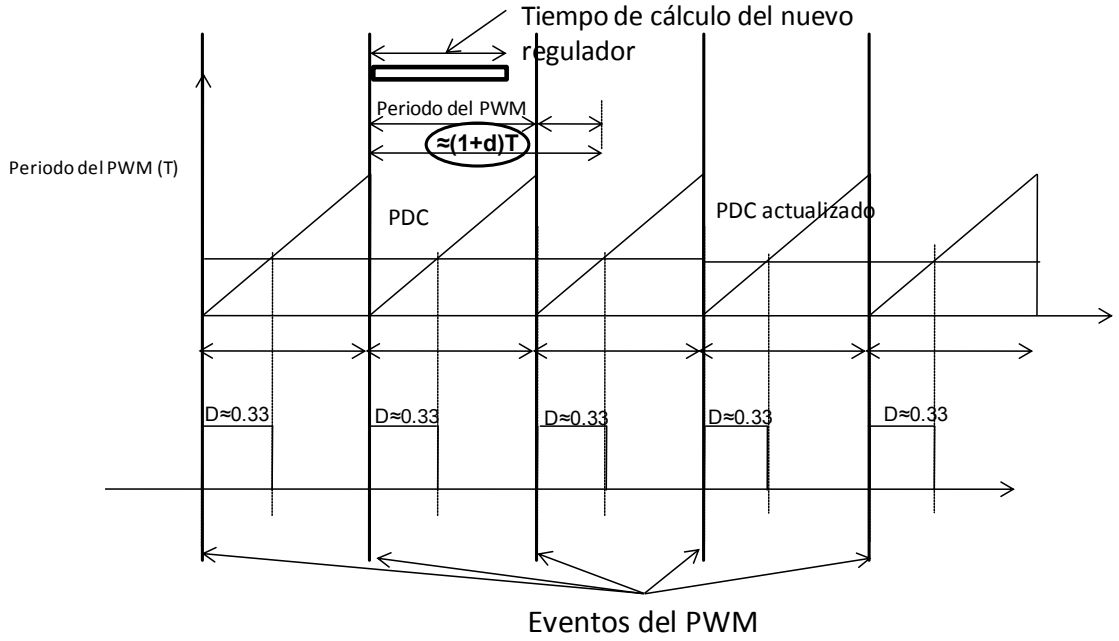


Figura 5-3 Instante de actualización del PWM

Por consiguiente el tiempo de retardo entre la toma de la medida de la señal a controlar, que coincide con el evento del PWM, y la actualización del ciclo de trabajo es de $1.33*T_{con}$, en este caso 1.33*3.33e-6 s.

Finalmente, este retardo restado a la fase medida, nos da una planta con el retardo teórico implícito del regulador digital a implementar y que en SmartCtrl sólo es posible introducir en la planta a importar. Mediante este retardo, se consigue que en el lazo del SmartCtrl esté contemplado el retraso de regulador que afecta muy significativamente a la estabilidad del sistema.

5.1.2 El sensor

El sensor de este lazo de control por tensión, es un divisor de tensión resistivo integrado en la placa *Buck*/*Boost*. Este divisor de tensión se encarga de adecuar la tensión de salida del convertidor, ya sea reductor o elevador para que a la entrada del dsPIC se alcancen 3.3V en caso de estar la tensión medida a su nivel máximo. Esto es así porque el fondo de escala del ADC integrado en el dsPIC es precisamente 3,3V, es decir, que cuando en el pin del ADC del dsPIC hay 3,3V, la conversión en digital pasa a ser "1" en todos sus bits (10 en este caso). Por lo tanto para las tensiones de salida máximas, 20V en el elevador y 5 V en los reductores, el sensor se encarga de pasar de esa tensión de salida al valor máximo asumible por el ADC del dsPIC, es decir 3,3V.

En principio no debería haber mucha más complicación, sin embargo experimentalmente se midió la respuesta frecuencial del sensor y se comprobó que a alta frecuencia no se comportaba como un simple divisor de tensión resistivo ideal, cuya ganancia es plana para todas las frecuencias.

Como ejemplo en la Figura 5-4 y Figura 5-5 se muestra la respuesta en frecuencia del sensor del reductor formado por un divisor de tensión resistivo cuyas resistencias son de 5Ω y 3,3Ω.

Este en teoría responde de la siguiente manera

$$V_{out} = {}^{5}/_{8,3} \cdot V_{in} = 0{,}6024 \cdot V_{in} \qquad \textit{Ec. 5-3}$$

Y por lo tanto la ganancia en dB:

$$20 \cdot log(0{,}6024) = -4{,}4dB \qquad \textit{Ec. 5-4}$$

En la siguiente gráfica se ve que para frecuencias bajas y medias la ganancia es precisamente de -4.4dB pero a partir de 30kHz la ganancia sube rápidamente.

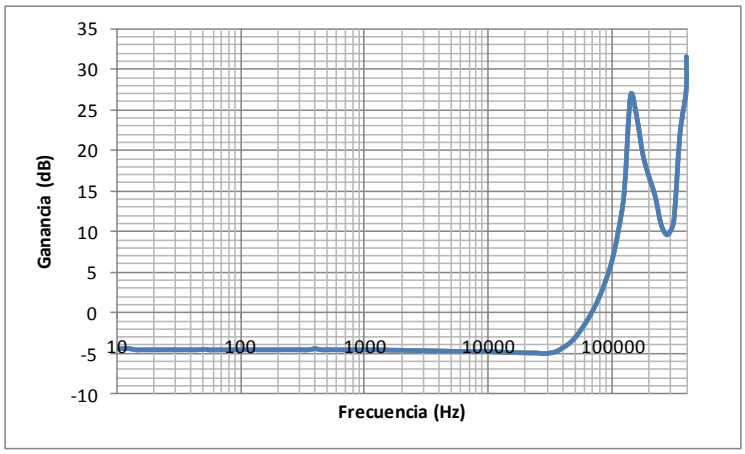


Figura 5-4 Ganancia medida del sensor

Para la fase el comportamiento a alta frecuencia también es particular.

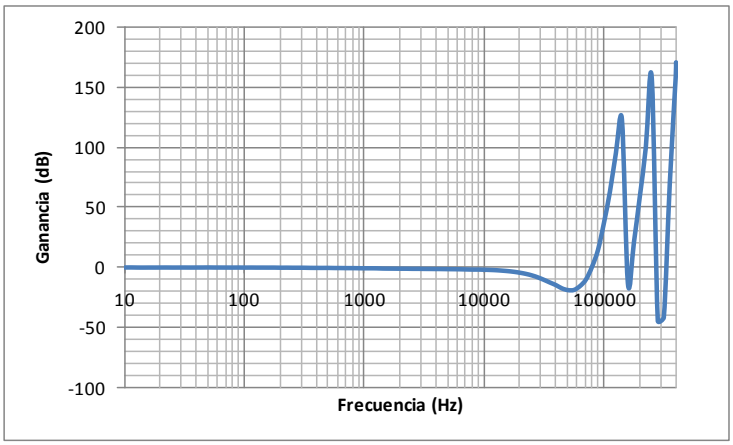


Figura 5-5 Fase medida del sensor

Este comportamiento a alta frecuencia no se ha introducido en SmartCtrl salvo en el caso de los reguladores tipo3, como se explicará en el apartado 8.3.3. Por tanto, en los resultados experimentales de PI y tipo2, la medida y lo que nos proporciona SmartCtrl en ganancia y fase difieren, ya que este comportamiento no se ha introducido en el lazo de SmartCtrl pero si existe en el montaje experimental. Sin embargo, su incidencia en el comportamiento del sistema es baja, ya que a dichas frecuencias la ganancia en lazo abierto del sistema es baja, aun considerando el efecto del sensor descrito.

Por otro lado el sensor se ha modelado con la opción de divisor de voltaje de SmartCtrl, pero no simplemente con la ganancia negativa correspondiente al divisor de tensión de la placa Buck/Boost, sino que se ha aprovechado para introducir en el lazo una ganancia

particular provocada por el dsPIC que se explicará con más detalle en la medidas experimentales (7.1.2.2.6). Esta ganancia introducida en el bloque de sensor de SmartCtrl, no pertenece al mismo sensor real, sino que es propio del programa del dsPIC, su ADC y su modulación PWM y por lo tanto es una ganancia introducida por este último y no por el sensor. Sin embargo se ha introducido en este punto para que una vez ahí, el diseñador pueda olvidarse de ella y diseñar el regulador pudiendo ver en SmartCtrl el diagrama de Bode de la ganancia en lazo abierto (Margen de Fase, MF y Frecuencia de cruce, Fc) tal y como saldrá en la medidas experimentales.

De un modo similar y como se explicará en los apartados 5.1.3 y 8.3.3, el sensor de SmartCtrl se usa para introducir otra ganancia necesaria para el caso de los reguladores de tipo3. Sin embargo de nuevo el objetivo es el de conseguir que una vez introducida esta ganancia el diseñador pueda olvidarse de ella y centrarse en la selección de MF y Fc del lazo.

5.1.3 Planta y sensor sumados para los reguladores tipo3

Como ejemplo, para tener en los resultados de SmartCtrl finales incluido el efecto de alta frecuencia introducido por el sensor, se han sumado la planta y el sensor medidos, de tal modo que del esquema general de la Figura 5-6, se pasa al que se ve en la Figura 5-7.

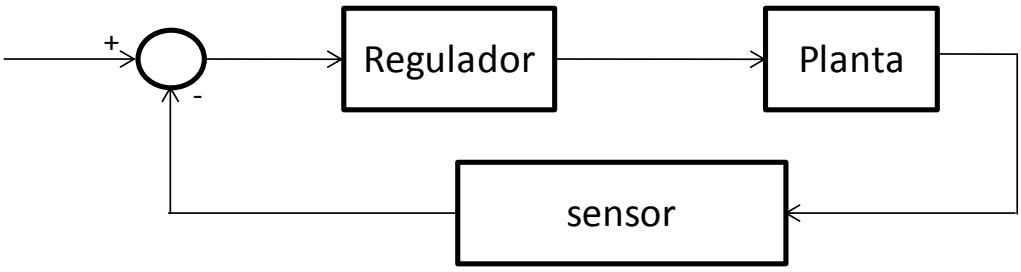


Figura 5-6 Esquema general del lazo

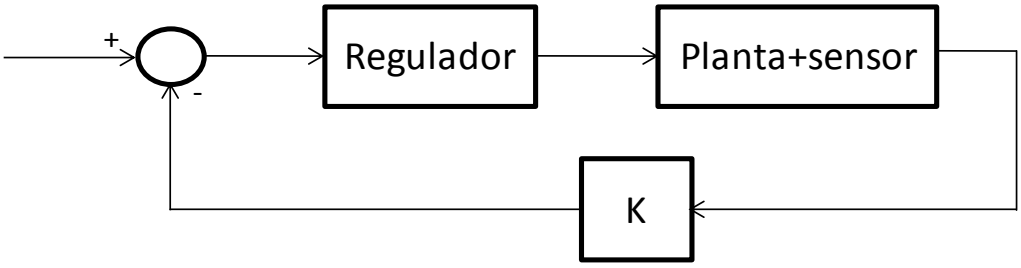


Figura 5-7 Esquema del lazo para el caso de los reguladores de tipo 3

Esto es posible gracias a que planta y sensor están en serie y se pueden sumar sus efectos para calcular el regulador. En este ejemplo, planta y sensor se suman, se ajustan y es este resultado el que se exporta a SmartCtrl. Este ajuste consiste en sumar 4dB a este resultado. Esto se hace así porque precisamente la ganancia del sensor proviene de:

$$^{5}/_{8,3} \approx 0{,}6309 \rightarrow 20 \cdot log(0{,}6309) \approx -4dB \qquad \textit{Ec. 5-5}$$

Así, se quita esta ganancia constante del sensor, dejando sólo en esta nueva planta el efecto de alta frecuencia.

Con ello el comportamiento a alta frecuencia del sensor sí está contemplado en el diseño en SmartCtrl, pero en este caso la planta introducida no es la real, sino una ficticia que viene de la medida de la planta real, más el retardo, más el sensor y más el ajuste de ganancia.

Esta situación hará que el valor de sensor introducido en el bloque de SmartCtrl tenga un valor en concreto que se detallará en los resultados experimentales de los tipo3 en el apartado 8.3.3.

Finalmente la planta importada en estos casos de tipo3, será la que se muestra a continuación.

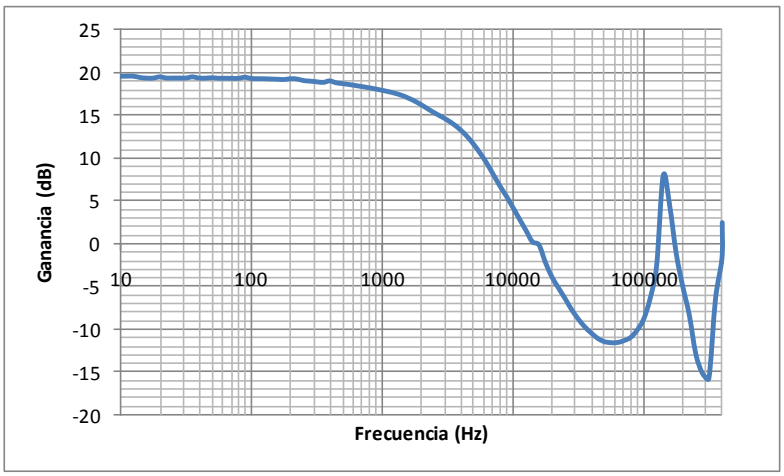


Figura 5-8 Ganancia de la planta importada para los tipo3

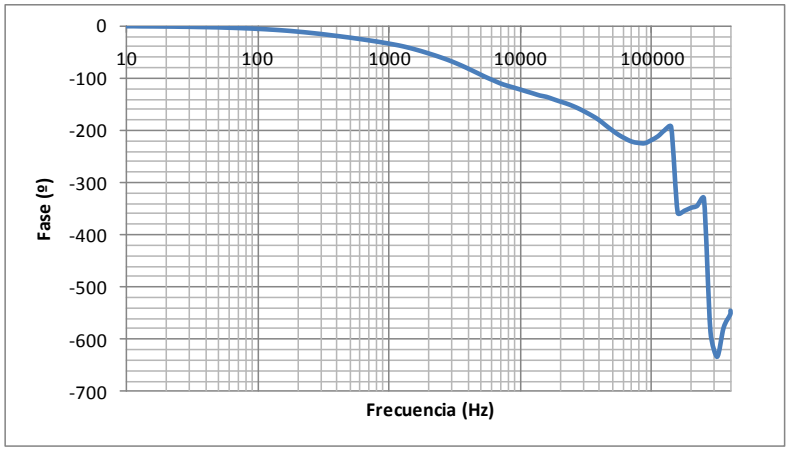


Figura 5-9 Fase de la planta importada para los tipo3

En estas figuras se ve que a alta frecuencia está sumado el efecto del sensor, pero sin embargo a medias y bajas la respuesta no tiene el efecto del sensor gracias al ajuste, ya que como se puede ver, la ganancia de planta empieza en 20dB que provienen de tener V_{in}=10V en un reductor.

$$Gain(f = 0) = 20 \cdot log(10) = 20dB \qquad Ec.\ 5\text{-}6$$

5.1.4 El tipo de regulador

SmartCtrl permite seleccionar entre reguladores de tipo2, 3, PI y polo simple. En este proyecto sólo se han usado el tipo2, 3 y PI. En esta parte del proceso de diseño simplemente se ha seleccionado uno de estos tres tipos de regulador.

En la Figura 5-10, puede verse la imagen donde se seleccionan y definen la planta, el sensor y el regulador.

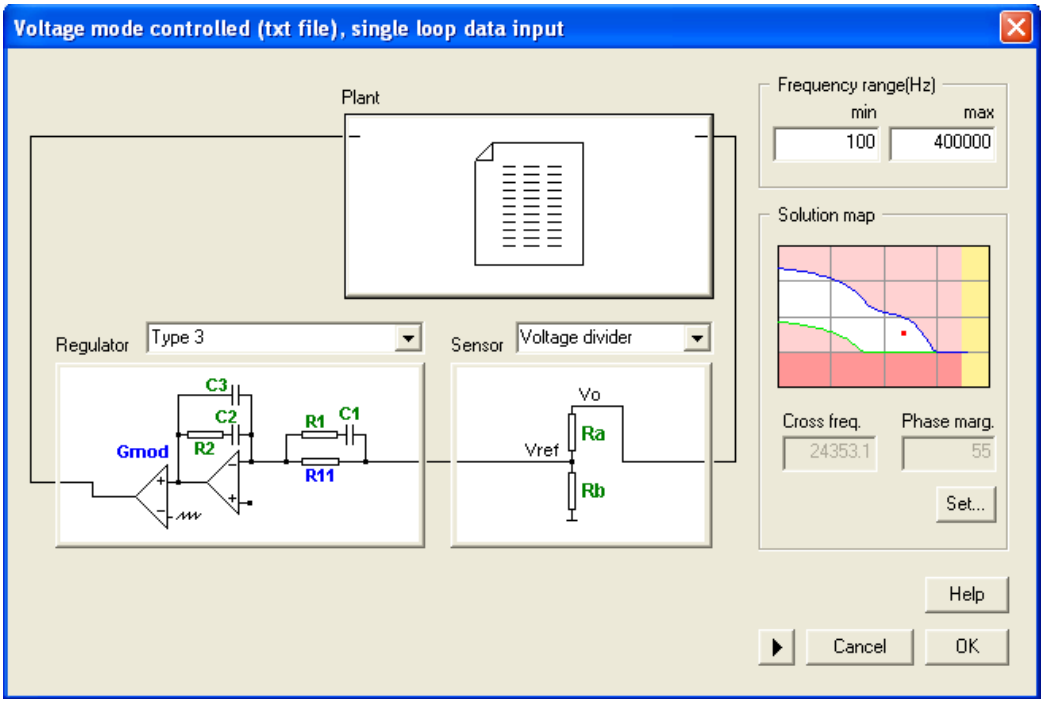


Figura 5-10 Ventana de selección y configuración de los bloques del lazo

5.1.5 Selección de frecuencia de corte y margen de fase en el mapa de soluciones

Una vez se han definido las partes integrantes del lazo, se tiene que la planta y el sensor están definidos y ya no se van a modificar más. Sin embargo del controlador solo se ha seleccionado el tipo de regulador y con el ello el orden del mismo. Para terminar de definir el regulador y con el ello el lazo, se debe seleccionar un valor de frecuencia de cruce (Fc) y otro de margen de fase (MF) para el lazo. En base a estos dos puntos, SmartCtrl dimensionará unos componentes del control analógico para obtener una respuesta en frecuencia, de tal modo que el lazo abierto cumple con las especificaciones de Fc y MF seleccionados en el mapa de soluciones. Además y lo que para este proyecto es más importante, ofrece al usuario diseñador, los coeficientes de la función de transferencia del regulador que cumple con las especificaciones seleccionadas. En la Figura 5-11, se ve la ventana que permite seleccionar un punto del mapa posible de soluciones.

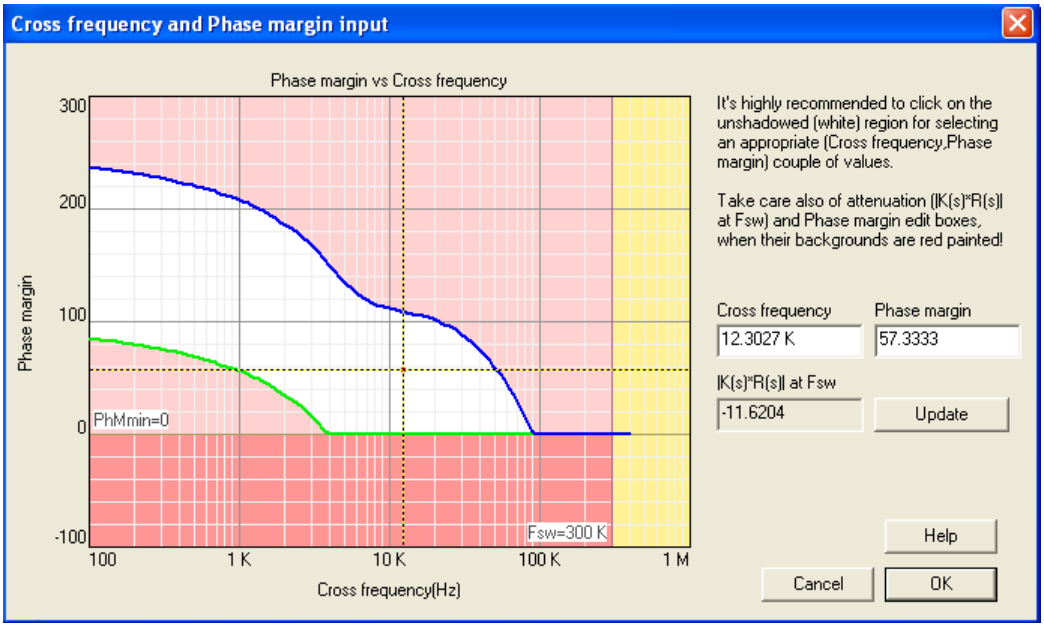


Figura 5-11 Selección de Fc y MF dentro del mapa de soluciones posibles

En este proyecto han sido implementados multitud de reguladores, tanto para medidas experimentales, como para simples pruebas, por lo que en este apartado no se comentará más de las distintas soluciones que hay dentro del mapa. Sin embargo es recomendable para el buen funcionamiento del lazo, seleccionar una frecuencia de cruce y margen de fase razonables. Sabiendo que para este proyecto la frecuencia de conmutación es de 300kHz, se aconseja una frecuencia de cruce como mínimo una década antes de esta frecuencia (30 kHz) y además se recomienda también al usuario seleccionar un margen de fase razonable que a priori puede ser en torno a 40º ó superior. Un caso razonable sería el que se ve en la Figura 5-11.

En cualquier caso, debe quedar claro que en este paso del diseño lo que se calcula es la función de transferencia (FDT) del regulador continuo que se quiere implementar en el control digital.

Para finalizar, en la Figura 5-12 se muestra la solución que ofrece SmartCtrl una vez se ha realizado el proceso de diseño descrito anteriormente. De él, se ha hecho especial uso de los diagramas de Bode, para comprobar la estabilidad del lazo, y la subventana "*Output result panel*", de donde se extraen los coeficientes de la FDT del regulador en continuo y que serán discretizados en pasos siguientes para poder implementar el regulador diseñado en el dsPIC. En la misma Figura 5-12, se puede ver cómo la fase está gravemente afectada por el retardo introducido al manejar muestras digitales en el proceso de cálculo del regulador. Como se ve, la fase de la planta decrece muy significativamente para altas frecuencias. Esto da una idea del error que se puede cometer si la planta no está debidamente ajustada e importada a SmartCtrl. De no hacerse así, el margen de fase definido por SmartCtrl estaría muy lejos del real, lo que afectaría a estabilidad del lazo sin que el diseñador se dé cuenta hasta que lo tiene implementado experimentalmente.

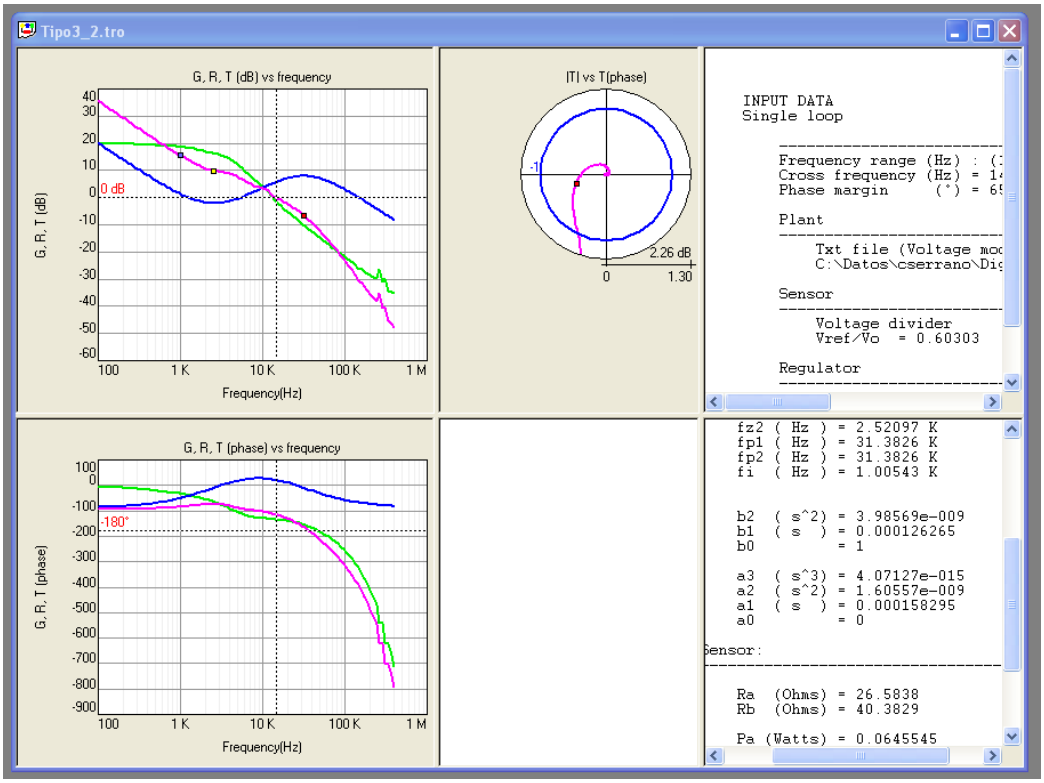


Figura 5-12 Solución de SmartCtrl

Por último, una vez seleccionado el punto del espacio de soluciones, SmartCtrl permite hacer cambios en la frecuencia de cruce y margen de fase cada uno por separado o de nuevo ambos juntos en el mapa de soluciones con mucha facilidad. Partiendo del primer diseño se pueden hacer cambios fácilmente sin tener que hacer de nuevo todo el proceso de selección de planta, sensor y regulador. Cada tipo de regulador tiene su propio método de recálculo: sea el PI *tuning* para el PI o el método del factor K para los tipo2 y 3. Esta funcionalidad del SmartCtrl ha resultado muy útil ya que permite modificar y diseñar nuevos reguladores de una forma muy rápida, sencilla y visual.

5.2 Discretizado con MATLAB

Llegados a este paso, se tiene el diseño de un regulador continuo realizado con SmartCtrl. Por lo tanto los pasos siguientes trataran de adaptar este regulador analógico al digital correspondiente alejándolo lo menos posible del original en el proceso de discretización, adaptación de los coeficientes y algoritmo de control. Se tiene que en este punto tenemos una FDT en continuo para el regulador, de la forma:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_0 s^m + \cdots + b_{m-1}s + b_m}{a_0 s^n + \cdots + a_{n-1}s + a_n} \qquad \text{Ec. 5-7}$$

Donde el orden de esta FDT depende del tipo de regulador, siendo el máximo orden 3, correspondiente al regulador de tipo3.

A continuación mediante un script de MATLAB, se introducen los coeficientes de la FDT en continuo, se discretizan por el método "*Tustin*" con un periodo de muestreo que se corresponde con el periodo de conmutación, y posteriormente se grafican ambas FDT

solapadas como se muestra en la Figura 5-13. En ella se ve que la FDT en continuo y discreto resultan coincidentes excepto al acercarse a la mitad de la frecuencia de conmutación.

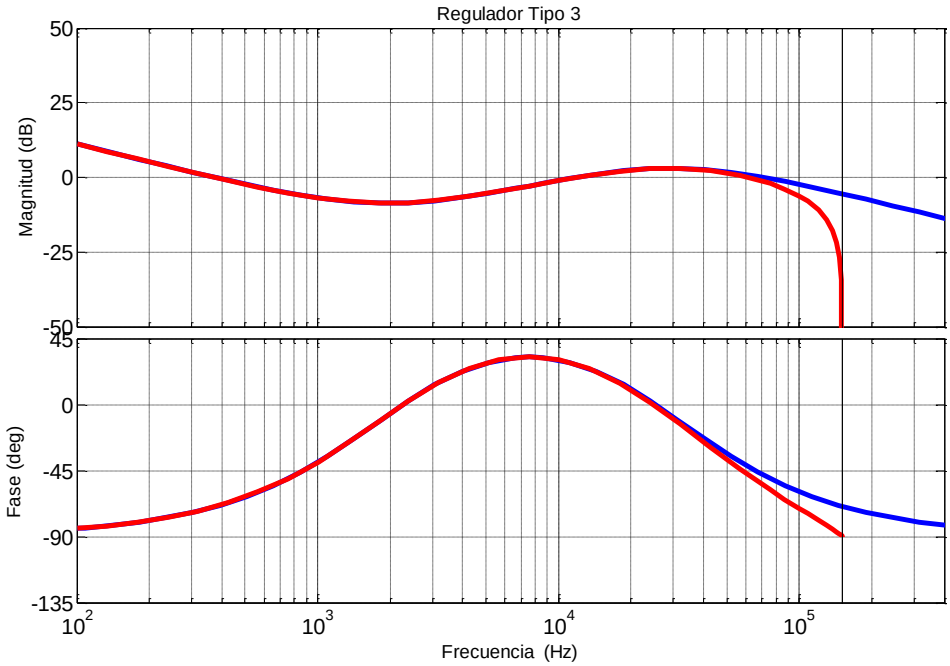


Figura 5-13 Ejemplo de un regulador continuo y el mismo discreto

Es en este momento cuando ya se obtiene una FDT discreta con la forma:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 z^m + \cdots + b_{m-1} z + b_m}{a_0 z^n + \cdots + a_{n-1} z + a_n} \qquad \textit{Ec. 5-8}$$

Donde los coeficientes desde b_0 hasta b_m y a_0 hasta a_n, serán usados por el dsPIC para realizar los cálculos del valor de control.

Por lo tanto como resumen en este apartado, se han conseguido los coeficientes de la FDT en discreto partiendo del continuo que se obtuvo del SmartCtrl.

5.3 Implementación del control mediante MPLAB

En este apartado es donde se tratarán los coeficientes de la FDT y se harán las modificaciones oportunas en el algoritmo de control. Para todos los tipos de control y distintos convertidores a gestionar se tienen la misma estructura de ficheros de código. Sin embargo, en función del tipo de control y convertidor o convertidores a gestionar, se harán unas u otras modificaciones. Todas estas modificaciones en función del convertidor se detallarán en el capítulo 5. Como resumen y para terminar el proceso de diseño de reguladores, se dirá que en esta parte del proceso se incluyen los coeficientes del regulador discreto calculados en el programa del dsPIC y se adecua el algoritmo de control a los valores de los coeficientes en caso de haberlos mayores o iguales a 1. La adecuación de los coeficientes se debe realizar puesto que se deben representar en un formato determinado (Q15, 16 bits con signo y representan sólo parte fraccionaria) que se detallará más adelante.

Puede darse el caso de que debido a la resolución finita del dsPIC, los coeficientes programados no cumplan la función de integrador por una mala interpretación del bit menos significativo de la parte fraccionaria introducida. Por lo tanto se comprueba si los coeficientes son leídos correctamente y de darse el fallo, se modifica ese bit menos significativo para que se cumpla la función de integrador propia de los tres tipos de control diseñados (En la práctica se resuelve el problema introduciendo la parte fraccionaria con todos sus decimales). Como ejemplo ilustrativo de este fenómeno se ha dejado en los resultados experimentales el lazo tipo2_2 detallado en el apartado 8.3.2.2.2.

Finalmente se implementa el regulador diseñado en el dsPIC, a través del software MPLAB IDE y el programador-*debugger* MPLAB ICD2. A partir de este punto se tiene ya el convertidor funcionado en lazo cerrado y se pasa a realizar las medidas experimentales sobre el mismo para comprobar que lo medido experimentalmente y lo implementado teóricamente son iguales.

5.4 Resumen de los pasos del proceso de diseño

Debido a la importancia del proceso de diseño de reguladores en este proyecto, se va a hacer un breve resumen en una tabla de los pasos seguidos en el proceso de diseño de los reguladores. En la práctica en condiciones normales este proceso de diseño conlleva menos de 20 minutos, por lo que mediante la tabla se resumen los pasos que debería realizar un usuario que quiera realizar su propio diseño.

PASO	DISEÑO	HERRAMIENTA	DESCRIPCION
1	Obtención de la planta	Analizador de ganancia y fase	Se mide experimentalmente la planta del sistema a controlar
2	Introducción del retardo en la planta	Excel	Se introduce un retardo en la medida experimental de la planta Retardo(f)=$t_{retardo}$*f*360
3	Introducción de los componentes del lazo	SmartCtrl	Se introduce en un modelo controlado por tensión en lazo simple los componentes del lazo
4	Selección de MF y Fc	SmartCtrl	Se fija sobre el mapa de soluciones el MF y la Fc
5	Modificación del MF la Fc (Opcional)	SmartCtrl	Si se considera oportuno se modifica el MF, la Fc o el factor K en caso de ser posible
6	Extracción de los coeficientes de la FDT continua	SmartCtrl	De la tabla de resultados de SmartCtrl se copian los coeficientes de la FDT (Bn…B0, Am…A0)
7	Discretizado de la FDT continua	MATLAB	Los coeficientes de la FDT continua se introducen en un Script de MATLAB para discretizar la FDT.
8	Introducción de los coeficientes de la FDT discreta en MPLAB	MPLAB	Se introduce la parte fraccionaria de los coeficientes en el fichero init.c y se modifica el PID.s para introducir las partes enteras
9	Depurado y primera vista del funcionamiento	MPLAB	Se ejecuta el programa para comprobar en una primera vista su funcionamiento y los valores leídos de los coeficientes en Q15
10	Comprobación de integrador en los coeficientes	Excel	Se comprueba si los valores que maneja el dsPIC en Q15 cumplen la función de integrador. Si no, se cambian para que lo haga
11	Programación	MPLAB	Programación final del dsPIC con coeficientes que en Q15 cumplen la función de integrador
12	Medida experimental del sistema implementado	Analizador de ganancia y fase	Se miden experimentalmente ganancia y fase de planta, regulador y lazo para comprobar que lo implementado se ajusta a lo teóricamente diseñado

Tabla 5-1 Pasos en el proceso de diseño

6 Implementación de controles digitales en dsPIC

En este capítulo se describirá el proceso y las dificultades que han aparecido a la hora de implementar el control digital. Partiendo ya de los coeficientes de la FDT obtenidos mediante el proceso descrito en el capítulo anterior, ahora el objetivo es introducirlos en el algoritmo de control para implementar el regulador. Primeramente se mostrará el flujograma general que sigue el programa con sus interrupciones en el flujo y a partir de ahí, se describirán los ficheros usados y su utilidad en la aplicación con sus diagramas de flujo particulares. Después se hará más hincapié en los distintos tipos de control, ya que estos varían el código del algoritmo en función del número de coeficientes que tiene la FDT discreta.

Es importante decir que la estructura general del código inicial fue aportada por el fabricante a través de notas de aplicación, llegándose a tener un control en lazo cerrado para otro dsPIC distinto al de este proyecto. Este fue adaptado para el dsPIC de este proyecto y a partir de él, se realizaron todos los cambios necesarios para obtener los reguladores diseñados mediante el proceso descrito en el capítulo 4. Por lo tanto el esquema general del programa se obtuvo de la web del fabricante, cuyo diagrama de flujo general es el siguiente.

6.1 Flujograma

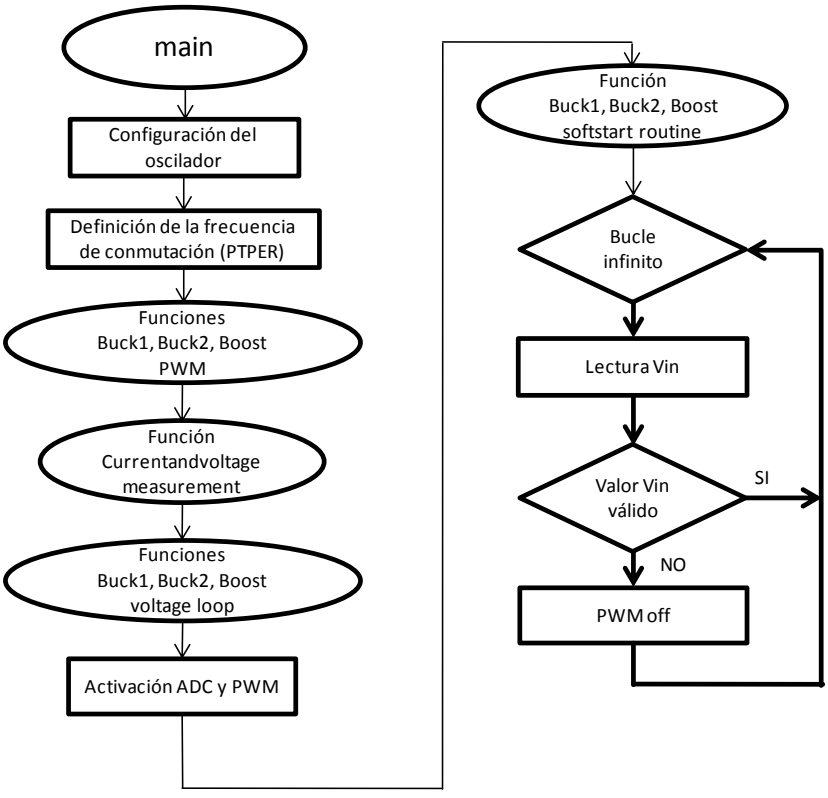


Figura 6-1 Flujograma del programa principal

Este es el flujo normal del programa. Donde en un primer momento se configura el reloj del mismo modo en que estaba en el programa del fabricante, es decir mediante el reloj interno del dsPIC. A continuación se fija el registro del PWM, PTPER, donde se puede fijar la frecuencia del PWM y con ello la de conmutación. Después se llama a una serie de funciones

donde se inicializan los PWM para los convertidores, el ADC para realizar las medidas de las variables a controlar y se introducen los valores de los coeficientes de la FDT en sus posiciones de memoria de datos, para que luego el algoritmo de control pueda acceder a ellos y realizar los cálculos. Tras todas estas funciones, el programa principal activa el ADC y el PWM, lo cual supone que la subrutina de atención a la interrupción del ADC, provocada por eventos en el PWM se puede activar y romper el flujo normal del programa. Después de esto se ejecutan otras funciones que se encargan de hacer que el valor de referencia a seguir por el control, arranque desde 0 hasta la referencia final mediante una rampa que dura 50 ms. Finalmente se entra en el bucle infinito donde el programa queda a la espera de que se produzcan interrupciones. Salvo excepciones, estas interrupciones son las del ADC, que al hacer su conversión y lectura de la variable a controlar, dispara una interrupción en la que se llama a la función PID. Es en esta función donde se calcula el nuevo ciclo de trabajo, para una vez obtenido, volver a la subrutina del ADC e introducir el nuevo valor de control en el registro PDC del PWM, que es el que define el ancho de los pulsos del PWM y con ello el ciclo de trabajo.

En la Figura 6-2, se muestran los flujogramas de la subrutina de atención a la interrupción del ADC y la de la función PID, que es la que realiza los cálculos del nuevo valor de control. Como se ve, estos flujos no tienen ni bifurcaciones ni bucles, ya que así se garantiza que el tiempo de cálculo es siempre el mismo y además conocido. Lo que como se comentó anteriormente, es importante para realizar los cálculos con las muestras de error y ciclo de trabajo correctas ya que estas afectan directamente al valor de la señal de control.

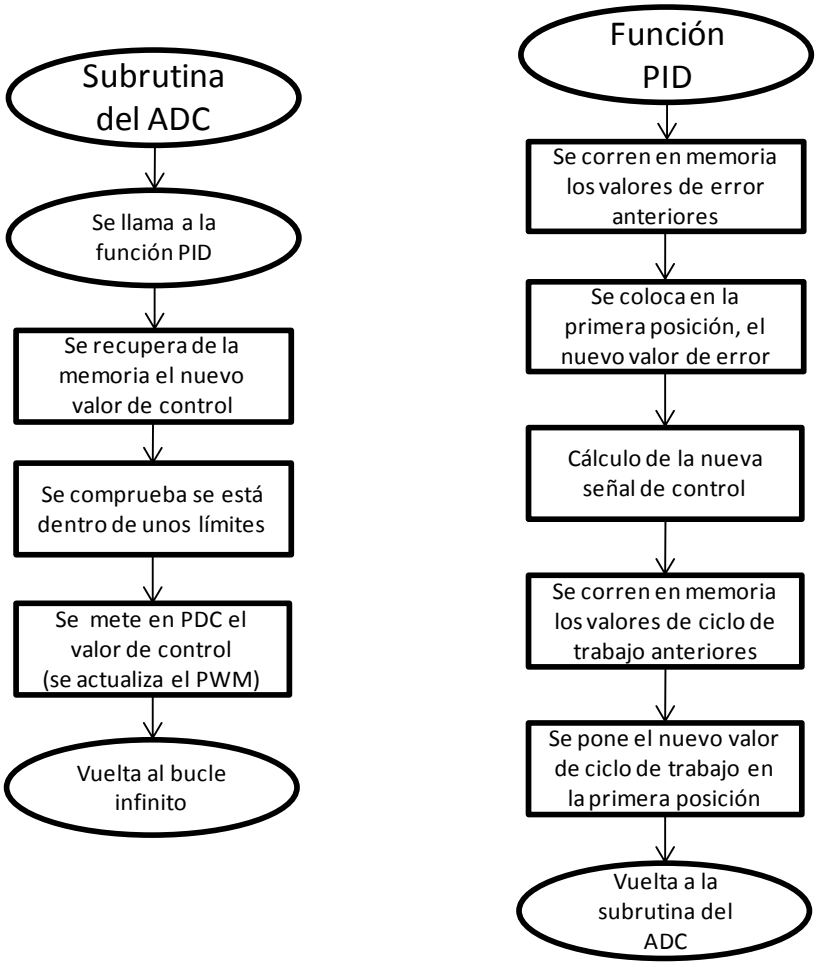


Figura 6-2 Subrutina de atención a interrupción (ADC) y subrutina de PID

6.2 Descripción de los ficheros y su función

Todos los ficheros descritos a continuación llevan en cabecera el fichero:

#include "p33FJ16GS502.h”

que es el fichero *.h propio del dsPIC que se está programando y que el fabricante publica para el uso de sus DSP.

6.2.1 Main.c

Se trata de un fichero en código C, donde está la función principal. Es en él, donde está lo primero que se ejecuta al iniciarse el programa. De manera más detallada que en el flujograma principal Figura 6-1, se dan las siguientes instrucciones.

Para empezar, se inicia incluyendo los ficheros:

#include "p33FJ16GS502.h"

#include "Functions.h"

Donde en el Functions.h simplemente se enuncian los nombre de las funciones que son llamadas desde la función main.

Se continúa ejecutando instrucciones que permiten que los programas del dsPIC, puedan ser depurados mediante la herramienta ICD2, ya que sincroniza ambos incluyendo el reloj interno del dsPIC. Además se indican unos valores numéricos que harán de límites para la tensión de entrada Vin. Una vez se ha configurado el reloj, lo siguiente que se define es la frecuencia a la que va a funcionar el PWM, es decir la frecuencia de conmutación. Se sabe que la resolución del PWM es de 1.04ns con lo que si se quiere tener un PWM de 300kHz el PTPER debe ser:

$$PTPER = \left(\frac{\frac{1}{300KHz}}{1.04ns} \right) = 3205 \qquad \textit{Ec. 6-1}$$

A continuación se realizan las llamadas a las funciones de configuración. De las cuales en esta descripción genérica están todas, sin embargo en los programas reales no, ya que no es posible poner juntos el reductor 2 y el elevador por hardware. Por lo tanto para cada caso solo están habilitadas unas u otras, pero no todas. En el caso de controlar conjuntamente reductor 1 y 2 estarán sólo las dos funciones correspondientes, al igual que en caso de controlar reductor1 y elevador. En caso de trabajar sólo con un convertidor, se tendrán sólo las suyas, es decir, las del reductor 1 o las del reductor 2 o las del elevador, depende siempre de cuantos y cuales convertidores se quieren controlar. Finalmente las funciones llamadas en esta parte son:

-Buck1Drive

-Buck2Drive

-BoostDrive

-CurrentandVoltageMeasurements

-Buck1VoltageLoop

-Buck2VoltageLoop

-BoostVoltageLoop

-Buck1SoftStartRoutine

-Buck2SoftStartRoutine

-BoostSoftStartRoutine

Antes de hacer las llamas a las SoftStartRoutine, se activan el PWM y el ADC previamente configurados en las funciones anteriores, con lo que a partir de este momento las interrupciones del ADC pueden darse y se darán periódicamente siguiendo el disparo provocado por eventos del PWM. (Cada ciclo de PWM, cada 2, cada 3, depende del caso y de la aplicación como configurar el PWM)

A continuación se configura el pin que permite a la carga del reductor 1, conducir o no. Esta función de la placa no ha sido usada en este proyecto, ya que han sido usadas cargas conectadas directamente a los conectores de salida de los convertidores, dimensionadas para que el convertidor funcione en MCC.

Finalmente se entra en el bucle infinito, (*idle loop*) a la espera de interrupciones. En el bucle en sí, se comprueba si hay nuevas lecturas de la entrada analógica correspondiente a V_{in}, y en caso de haberla se comprueba si está dentro de unos límites mínimo y máximo. En caso de estar fuera, deshabilita los PWM de los convertidores, con los que estos se apagan. (La salida de los reductores será 0 y la del elevador será Vin)

6.2.2 Init.c

En esta función escrita en C, además de contener todas las funciones que son llamadas desde el main para configurar PWM y ADC, se definen variables, se les asignan valores en formato Q15 y se reservan espacios de memoria X e Y. Por lo tanto, se va a fraccionar su explicación en varios puntos debido a la importancia que esto tiene para entender el resto de funciones y ficheros.

6.2.2.1 El fichero dsp.h

La función init.c empieza incluyendo:

#include "p33FJ16GS502.h"

#include "Functions.h"

#include "dsp.h"

Donde el dsp.h, define variables y estructuras necesarias para que el dsPIC pueda hacer cálculos de tipo DSP. Este fichero lo aporta el fabricante para que el programador pueda usarlo para programar sus dsPIC y realizar cálculos complejos y en coma flotante. Sobre este fichero no se han realizado cambios, pero es importante entender para que sirve y que aporta ya que es muy complejo. En principio sirve sólo para poder realizar cálculos complejos por lo que no es aconsejable cambiar nada de su código, ya que el fabricante así lo ofrece, supuestamente optimizado para ser usado con sus dsPIC. En él, se definen el tipo de variables tPID, que se usarán en el programa y se define el formato Q15. Estos dos temas se tratarán a continuación

6.2.2.2 El formato Q15.

El formato Q15 es el formato que maneja el dsPIC para realizar sus cálculos en coma fija. El dsPIC normalmente maneja variables de 16 bits, que en formato Q15, supone 15 bits para una parte fraccionaria y el último bit para el signo.

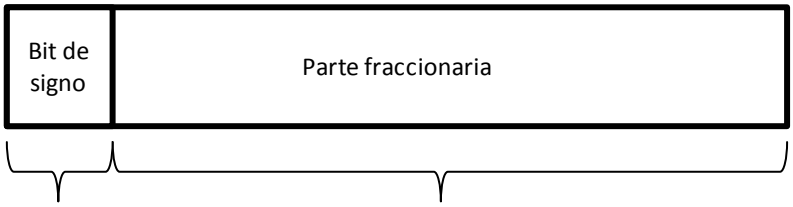


Figura 6-3 Formato Q15

Cuando se habla de parte fraccionaria, se refiere a que estamos trabajando con números entre 0 y un número próximo a 1 pero menor. Entre un número en Q15 y el siguiente hay una diferencia real de:

$$2^{-15} = 3.0517578 \cdot 10^{-5} \qquad \textit{Ec. 6-2}$$

Es decir, que el dsPIC no es capaz de ver diferencia entre 0.99999 y 0,9999695, ya que el mayor valor que puede manejarse en Q15 es:

$$1 - 2^{-15} = 0.999969482421875 \qquad \textit{Ec. 6-3}$$

Donde

$$0.999969482421875 < 0.9999695 \qquad \textit{Ec. 6-4}$$

Por lo tanto entre 0.9999695 y 0.99999, el dsPIC no interpreta diferencias ya que tiene resolución finita.

Finalmente se tiene que el número decimal más alto que corresponde en bits a 7FFF hexadecimal, es 0.999969482421875 y superiores sin llegar a 1.

Para los números negativos, se hace el complemento a 2 y quedan los mismos problemas de resolución que con números positivos, teniendo que el número más negativo, el correspondiente a 8000 hexadecimal es -0.999969482421875 e inferiores sin llegar a -1.

6.2.2.3 Las variables tPID

Estas variables se definen según la estructura:

```
typedef struct {
    fractional* Coefficients
    fractional* Control;
    fractional ControlOutput;
    fractional MeasuredOutput;
    fractional ControlReference;
} tPID;
```

En esta estructura como se puede ver hay 5 campos a completar, donde cada uno de ellos corresponde a una posición de memoria de datos donde **Coefficients* y **Control* son

punteros a otras zonas y *ControlOutput*, *MeasuredOutput* y *ControlReference* son posiciones de memoria donde se guarda información. La siguiente figura muestra como sería esa zona de memoria donde está la variable tPID.

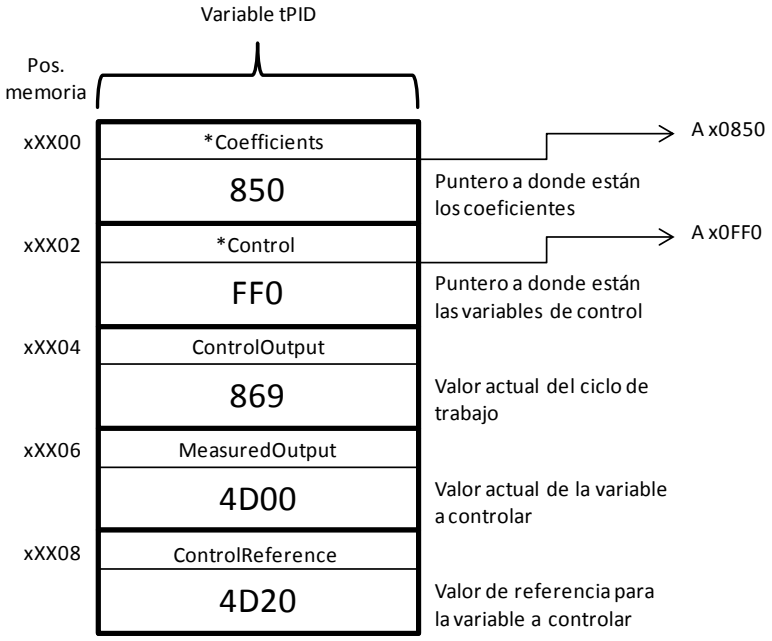


Figura 6-4 Esquema básico en memoria de la variable tPID

6.2.2.3.1 *Coefficients

Fractional **Coefficients* es un puntero a una zona de la memoria X, donde se almacenarán números de tipo fraccionario Q15. Estos números, son los coeficientes de la función de transferencia si estos están entre 0 y menor de 1 o sólo su parte fraccionaria en caso de ser números mayores que 1. Está en forma de puntero porque dependiendo del tipo de regulador, se tienen más o menos coeficientes y por lo tanto más o menos posiciones de memoria deben ir a continuación de aquella a la que apunta el puntero **Coefficients*, para poder ubicar en memoria todos los coeficientes. De esta manera, los coeficientes son accesibles para luego poder usarlos en los cálculos del control.

6.2.2.3.2 *Control

Fractional **Control*, es como en el caso anterior un puntero a una zona de memoria, que en este caso es memoria Y, donde se almacenan números en Q15. En este caso, a partir de la dirección de memoria donde apunta **Control*, están los valores de error y sus muestras anteriores y los valores de ciclo de trabajo y sus muestras anteriores. Todos estos quedan juntos porque más adelante al hacer los cálculos del regulador se van recorriendo estas zonas de memoria para poder usar los números en los cálculos. Como en el caso anterior, el número de posiciones a partir de aquella a la que apunta **Control*, dependen del tipo de control a implementar. Hay que recordar que para realizar los cálculos de tipo DSP o MAC, uno de los multiplicandos debe estar en la zona Y, como se comentó ya en el apartado 3.1.2. En este caso serían estos lo que están en dicha zona de memoria, los que corresponden al error y sus muestras anteriores y al ciclo de trabajo y sus muestras anteriores.

6.2.2.3.3 ControlOutput

ControlOutput es una posición de memoria donde el programa almacena la última señal de control. Esta es fácilmente accesible ya que se accede a ese valor de ciclo de trabajo mediante direccionamiento directo. Es desde aquí desde donde se extrae este valor de control, para meterlo en el registro PDC, que define la anchura del pulso PWM.

6.2.2.3.4 MeasuredOutput

MeasuredOutput es muy similar la variable anterior, pero en este caso lo que se almacena es el valor actual de la variable a controlar. Este valor en régimen permanente, será el que corresponda al valor de la variable a controlar cuando está alcanza su valor estable.

6.2.2.3.5 ControlReference

De nuevo una zona me memoria como las anteriores. En este caso, aquí se almacena el valor de referencia que se debe alcanzar. Como se dijo anteriormente, este valor al principio es cero y se va incrementando hasta alcanzar el valor final de referencia, en una rampa de subida programada en 50 saltos de 1ms cada uno. Una vez alcanzados los 50ms, se tiene la referencia final que está seleccionada para tener a la salida la tensión deseada, ya sean 3,3V, 4,5V ó hasta 5V en el reductor, y en el elevador desde la Vin hasta 20V.

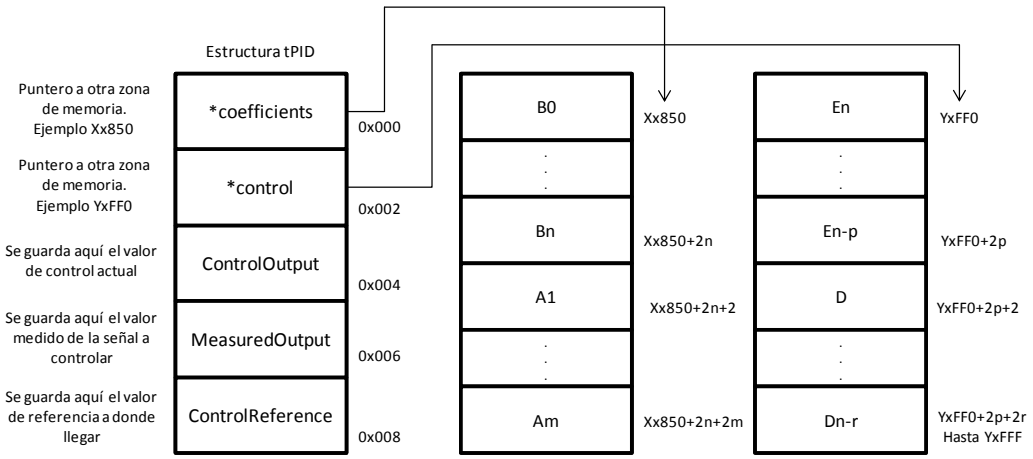


Figura 6-5 Disposición en memoria de las variables tPID

En la Figura 6-5, se ve el esquema general de cómo se organizan estas variables tPID.

Aparte en el apartado 4.4.1, se muestra la Figura 4-9, donde se ve un ejemplo real de variable tPID mientras se hace un depurado del programa mediante la ventana *"File Registers"* del MPLAB IDE.

6.2.2.4 Funciones del Init.c

Una vez se ha visto todo esto, se volverá a la descripción general del Init.c. Tras incluir el fichero dsp.h, se declaran las primeras variables de tipo tPID, descritas anteriormente. Estas son las variables Buck1VoltagePID, Buck2VoltagePID, BoostVoltagePID, que declaran así todos los campos que se usarán más adelante en los cálculos del controlador. Después se reservan zonas de memoria tanto en X como en Y. En estas reservas es donde irán los valores de error y ciclo de trabajo con sus muestras anteriores y las partes fraccionarias de los coeficientes de la

FDT. Hay que recordar que en variables tipo tPID, hay dos punteros que apuntan a las cabeceras de estas zonas, y por lo tanto la reserva hay que hacerla igualmente para que queden disponibles las posiciones deseadas, y no las ocupe el compilador con otras cosas ya que afectaría directamente a los cálculos. Es muy importante dar a estas zonas de memoria el tamaño necesario y en ningún caso menos del necesario. Si no se hiciera, al hacer los cálculos, se extraerían datos de zonas de memoria con valores desconocidos o en el caso de la memoria en Y, que queda al final del banco de memoria, haríamos una solicitud de extracción de datos de memoria en una posición mayor de la máxima que tiene la memoria del dsPIC para datos. Lo que se ha comprobado no da error al compilar, pero si bloquea la ejecución del programa.

Seguidamente se define la parte fraccionaria de los coeficientes de la función de transferencia, que serán números entre 0 y un número próximo a 1 pero menor, que se obtuvieron del proceso de diseño con SmartCtrl, MATLAB y que son ajustados a Q15 (dejando sólo su parte fraccionaria). Estos serán más o menos coeficientes en función del tipo de regulador, donde se definirán como A0 hasta An para los del denominador de la FDT discreta y B0 hasta Bn, para los del numerador. Por último en esta zona de definiciones, se definen los valores numéricos de las referencias a seguir, que serán dos distintas para poder hacer escalones de referencia y unas variables de tipo entero, con algunas de ellas definidas como externas, ya que se usaran en otras funciones. Estas son, una variable para contar el número de interrupciones del timer (para hacer la rampa de subida en la referencia de 50ms), dos referencias y un flag que se usarán para cambiar de una referencia a otra mediante el botón de la placa *starter board*, usándose el flag para saber en cuál de las dos referencias está el programa actualmente. Tras esto, viene ya el código de configuración de PWM, ADC y demás, mediante las funciones que son llamadas desde el *main* y que están definidas en el Init.c

6.2.2.4.1 Buck1Drive

En estos apartados se describirán las configuraciones en general, para ver los valores concretos véanse los anexos.

En general estas funciones definen y configuran el modo de funcionamiento del generador PWM, cuyo hardware ha sido descrito en el apartado 3.1.3. En la Figura 6-6 se muestra sobre el flujograma principal la ubicación de las llamadas a las funciones Buck1Drive, Buck2Drive y BoostDrive descritas a continuación.

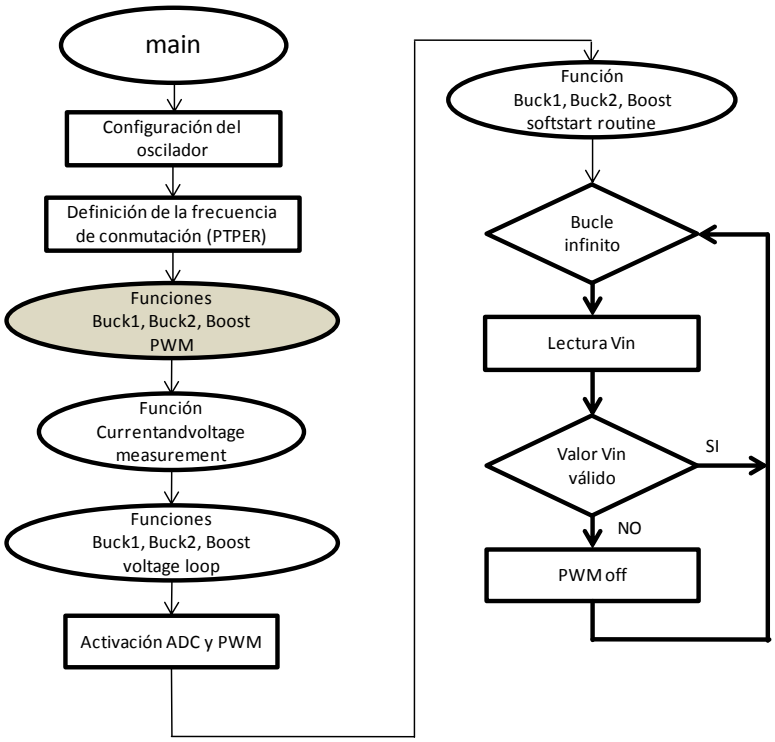


Figura 6-6 Llamada a funciones Buck1Drive, Buck2Drive y BoostDrive desde el programa principal

En primer lugar se configuran los puertos que se van a usar como PWMxH y PWMxL. Es decir los pines 25 y 26 funcionarán como tales, no como GPIO, (*General Purpose I O*) y en modo complementario, ya que ambos reductores son síncronos. Se continúa definiendo el *dead-time* como positivo, para confirmar que los dos MOSFET del reductor síncrono no conducen al mismo tiempo lo que evita provocar un cortocircuito en Vin, definiéndose posteriormente la anchura de este *dead-time* para ambos flancos. Después se configura para que la actualización del ciclo de trabajo, se dé en el siguiente periodo y no cambiando el ancho del PWM en el mismo instante en que se finaliza el cálculo del nuevo valor de control. Se sigue seleccionando el PWM con frecuencia de conmutación por PTPER, que para esta aplicación es mucho más sencillo e intuitivo que otras disponibles en el dsPIC, más enfocadas para convertidores multifase. Por último en este bloque se deshabilitan las entradas de fallos, que no han sido configuradas.

En el tramo final se definen el número de eventos del PWM que se tienen que dar para que se dé la interrupción por eventos del PWM y que dispara la interrupción del ADC. Este valor ha sido puesto de dos formas distintas. La primera es en caso de funcionar el reductor 1 en paralelo con otro de los convertidores, en cuyo caso se ha puesto que el PWM1 dispare la interrupción del ADC cada 2 ciclos de PWM, permitiendo gestionar al dsPIC, los dos convertidores alternativamente. Y el segundo caso es aquel en el que funciona sólo el reductor 1, por lo que la interrupción del ADC es disparada cada ciclo de PWM. En realidad el reductor 1 no se ha hecho funcionar sólo mucho, ya que es el siempre está disponible por hardware para funcionar junto a los otros y por lo tanto en general, cuando este funcionaba lo hacía junto a otro. Para funcionar en solitario se han usado el reductor 2 y el elevador, que son los que permiten hacer las medidas experimentales de respuesta frecuencial, ya que se puede abrir el lazo sin romper pistas, ya que disponen de jumpers para ello. Seguidamente se define el número de ciclos que se espera para hacer la primera interrupción. En relación con lo anterior,

si funciona en paralelo con otro, estos deben alternarse, por lo que uno de ellos debe esperar un ciclo y el otro ninguno. Siendo el caso de reductor 1 el que no espera ningún ciclo y dispara la interrupción del ADC en el primer evento del PWM, siendo el otro convertidor el que se retarda un ciclo. Por lo tanto finalmente se inicializa el ciclo de trabajo con el valor mínimo de anchura de PWM1 mediante el PDC1 y se fija el retardo entre el primer evento del PWM y la primera interrupción en 0 mediante el TRIG1, para que la interrupción surja inmediatamente después del primer evento de PWM.

Con todo esto el PWM del reductor 1, queda inicializado.

6.2.2.4.2 Buck2Drive

Análogo al anterior pero ahora con todas las configuraciones de los registros referidos al PWM2. Debe tenerse en cuenta que este reductor sí se ha usado mucho en solitario, ya que permite medir su respuesta en frecuencia, por lo que en este caso las interrupciones del ADC se dan cada evento de PWM. En caso de funcionar con el reductor 1, las interrupciones se dan cada dos eventos y empiezan tras esperar un ciclo, lo que permite que la primera interrupción y con ello el primer cálculo del control sea para el reductor 1 y partir de ahí se alternan uno y otro.

6.2.2.4.3 BoostDrive

Muy similar a los anteriores, pero en este caso para configurar el PWM3, que será independiente y no complementario, ya que sólo se usará el PWMxH. Por lo demás es análogo el reductor 2, ya que en el elevador también se puede abrir el lazo mediante un jumper para hacer las medidas de respuesta frecuencial.

6.2.2.4.4 CurrentandVoltageMeasurements

En la figura siguiente se muestra que parte del flujograma principal es la que corresponde a la función CurrentVoltageMeasurement, donde una vez están configurados los registros del PWM, se pasa a los del ADC y los de las interrupciones.

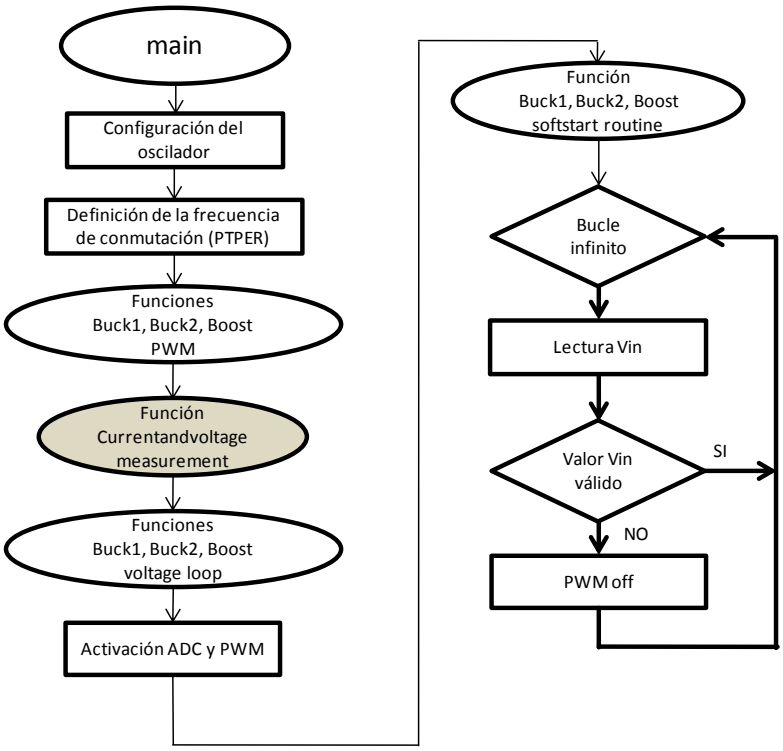


Figura 6-7 Llamada a función Currentand Voltage Measurement desde el programa principal

Los principales registros de configuración del ADC son el ADCPC y ADPCFG. En el ADPCFG se selecciona que pines del dsPIC funcionan como entradas analógicas. Los pines 2 y 3, que corresponden a AN0 y AN1, son para la corriente y tensión respectivamente del reductor 1. Los pines 4 y 5, que corresponden con AN2 y AN3, son los pines de corriente y tensión respectivamente del reductor 2 o del elevador según el caso, ya que se recuerda que ambos comparten pines del dsPIC, debido al hardware empleado. Por último el pin 6, AN4, corresponde a la tensión de la entrada a los convertidores y que es común a los tres. En cualquier caso, todas estas patas del dsPIC se configuran como entradas analógicas mediantes el bit correspondiente del ADPCFG.

Del registro ADCPC0, que controla las entradas AN0-AN1 y AN2-AN3 se habilitan las interrupciones del ADC. Mediante este registro se interconectan las interrupciones y lecturas del ADC, con los eventos del PWM. Es decir, es aquí donde el programador fija que la interrupción del ADC se dispara con el generador PWM1, el PWM2 ó el PWM3, teniendo en cuenta de nuevo que el 2 y 3 nunca coexisten por lo que o se ha puesto uno u otro, y lógicamente en cada caso el suyo. Es decir, AN0 y AN1 (corriente y tensión de reductor 1), dispararán la interrupción con el generador PWM1. Y AN2 y AN3 (corriente y tensión del reductor 2 o del elevador), dispararán con el generador PWM2 ó 3 según el caso. En esta inicialización se resetean los bits que indican que se ha dado una interrupción, tanto en los registros de ADC, como en los de interrupción, aunque en principio no tienen por qué estar dadas. Sin embargo en la inicialización se ponen a 0, ya que estos bits se ponen a 1 por hardware cuando se da la interrupción y a 0 por software, con lo que así en la inicialización se garantiza su reseteo.

En el registro de interrupciones IEC6 (*Interrupt Enable Control*) se habilita la interrupción de los pares 0 y 1 de las entradas analógicas como interrupciones, es decir, se

habilitan las interrupciones de las entradas analógicas de los convertidores. A continuación, es con el registro IPC27 donde se le asigna la prioridad a la interrupción.

Con todo esto vemos los tres factores implicados en la sincronización de PWM, ADC e interrupción para que el cálculo del algoritmo de control se haga donde interesa. Por un lado el PWM sincroniza todo con sus eventos, el ADC dispara su interrupción por que se dan los eventos del PWM correspondientes y es la interrupción la que trunca el flujo normal del programa, para que después de una lectura de las variables a controlar, se realice el cálculo del nuevo valor de control, que se hará efectivo en el siguiente ciclo de PWM. Con todo esto, un convertidor funcionando solo, cada periodo de PWM hace una lectura de hasta dos variables a controlar (tensión y corriente), realiza el cálculo del algoritmo de control y actualiza el valor PDC que fija la anchura del ciclo de trabajo. Este nuevo valor, en el caso de este proyecto se hará efectivo en el siguiente periodo de PWM, porque así fue configurado el registro correspondiente del generador PWM. Una vez se ha hecho efectivo el último valor calculado, de nuevo se estará realizando todo el proceso para el siguiente periodo. Por lo tanto, un periodo de PWM, conlleva todo el proceso de cálculo del control y actualización.

Por otro lado y para terminar, es en esta función donde se ha programado la interrupción provocada por el botón de la placa *starter board* (interrupción externa1 del dsPIC), para que se haga un cambio en la referencia, provocando un escalón en la tensión de salida, muy útil para ver la respuesta temporal del conjunto. Por hardware, el botón está conectado al pin 14, por lo que se ha remapeado este como entrada a interrupción externa 1. Como con las otras interrupciones, está en la inicialización se pone a 0 (a 1 se pone por hardware) y mediante IEC1 se habilita como interrupción externa 1.

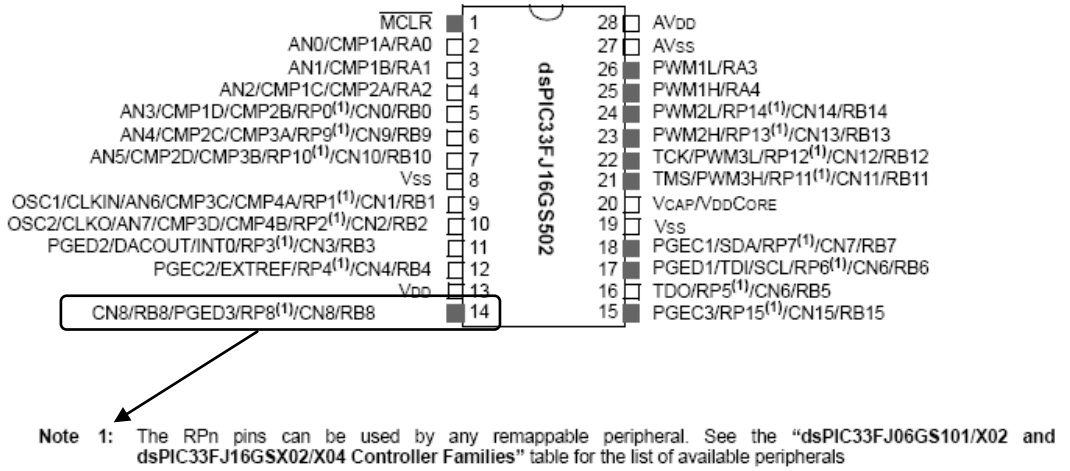


Figura 6-8 Pines del dsPIC33FJ16GS502, pin 14 remapeable, imagen extraída de [4]

6.2.2.4.5 Buck1VoltageLoop

El fichero init.c prosigue con las funciones VoltageLoop. En este caso con la del reductor 1, donde lo que se hace es obtener de la memoria las direcciones de los punteros **coefficients* y **control* que apuntaban a las zonas de memoria donde están los coeficientes y los valores de error y control. Estos, posteriormente serán usados en el algoritmo de control. Una vez se tiene la dirección que apunta a estos, se limpian estas zonas de memoria mediante la llamada desde aquí a la función PIDInit y después se llenan con los valores A0 hasta An y B0 hasta Bn que se definieron antes como se comentó en el apartado 6.2.2.4.

Se recuerda que estos valores son los números que se obtuvieron de SmartCtrl y que se modificaron para que quedasen como números entre 0 y un número menor que 1. Es ahora cuando el programa pasa esos números fraccionarios, a Q15 en hexadecimal de 16 bits. Previamente se comprueba que ninguno está fuera del rango válido en Q15 y en caso de salirse, el programa entra en un bucle infinito del que no se sale ya que recordemos que las interrupciones del ADC están configuradas, pero el PWM aún no ha arrancado y por lo tanto no hay eventos que lo provoquen.

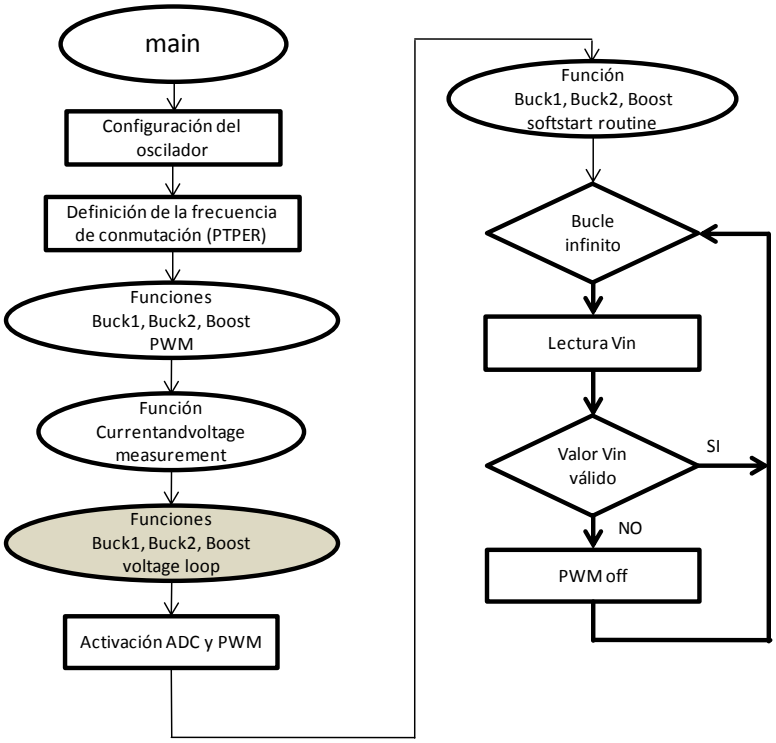


Figura 6-9 Llamada a funciones Buck1VoltageLoop, Buck2VoltageLoop y BoostVoltageLoop desde el programa principal

Tras introducir los coeficientes, los demás campos de las variables tPID, se inicializan a 0 salvo la referencia que se pone a un valor mínimo que es distinto de 0 y que posteriormente se irá incrementando en una rampa de subida hasta la referencia final. De nuevo en la Figura 6-9 se ve como se sigue la secuencia de definiciones del programa principal dirigiéndose al bucle infinito del programa.

6.2.2.4.6 Buck2VoltageLoop

Es equivalente al anterior pero para el reductor 2. Tiene el mismo número de coeficientes ya que para todos los convertidores que funcionan conjuntamente, se tiene la misma función que calcula el control PID (misma función de cálculo, implica mismo número de coeficientes y por tanto mismo tipo de control). Lo que sí se puede cambiar son los coeficientes, que pueden ser iguales lo cual es normal, ya que se trata de un reductor idéntico. El tipo de regulador no se puede cambiar con este código, entre un convertidor y otro cuando funcionan juntos.

6.2.2.4.7 BoostVoltageLoop

Análogo a los anteriores para el elevador. Mismo número de coeficientes y distintos o iguales coeficientes, en este caso hay que tener en cuenta que la planta es un elevador y por lo tanto tiene una dinámica distinta al reductor, con las implicaciones que esto tiene en el lazo y su estabilidad. Sin embargo se recuerda que si funciona conjuntamente con un reductor, la función PID, es la misma para los dos, aunque cada uno con sus coeficientes. Es decir, no puede haber uno de tipo PI para el reductor y otro tipo3 para el elevador a la vez.

6.2.2.4.8 Buck1SoftStartRoutine

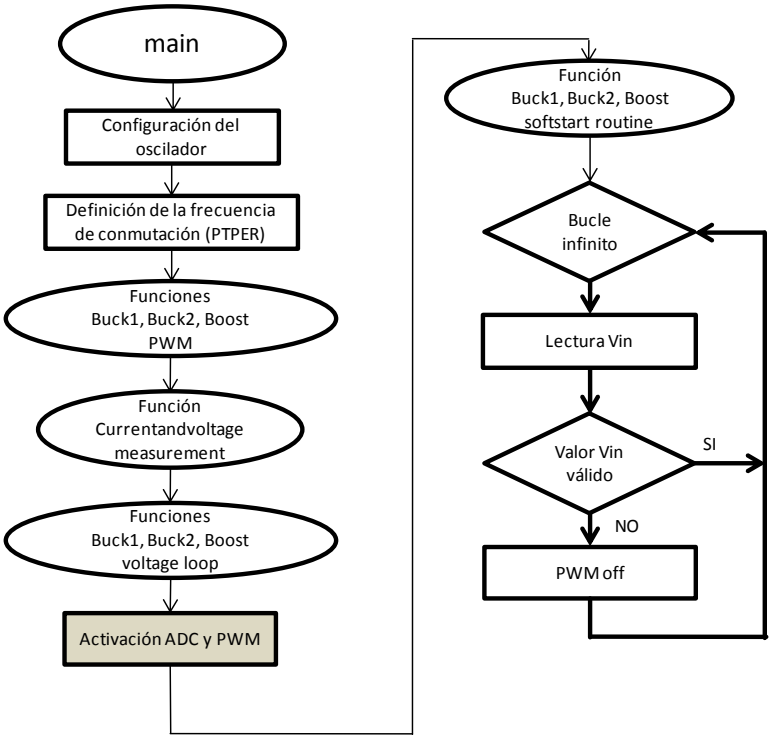


Figura 6-10 Activación de lecturas e interrupciones de ADC y PWM desde el programa principal

Hay que recordar que estas funciones son llamadas por la función *main*, pero es justo antes de esta función, cuando de activan el PWM y el ADC. Por lo que antes de que se ejecute esta función, las interrupciones del ADC pueden darse, saltado de esta rutina a la del ADC.

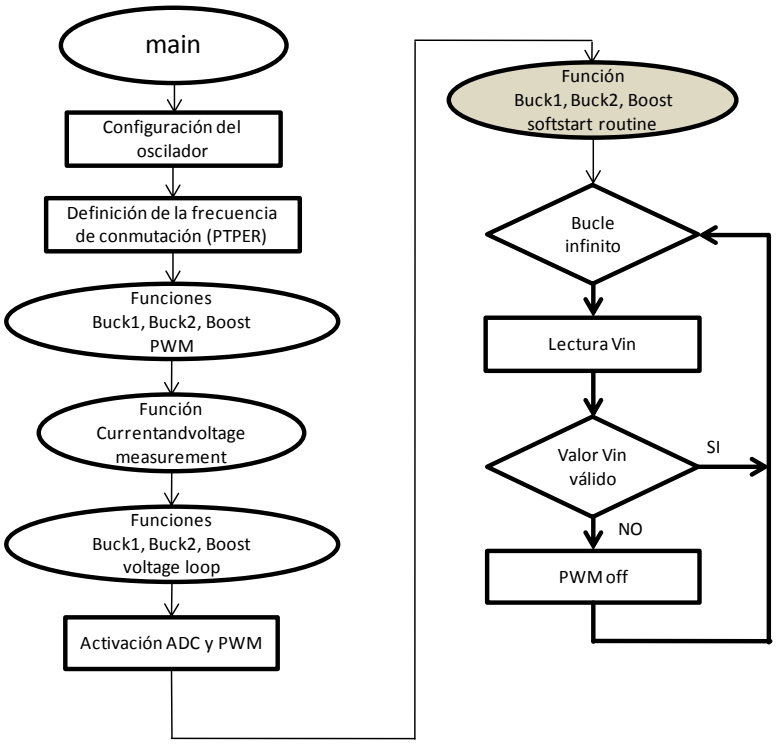


Figura 6-11 Llamada a funciones Buck1SoftStartRoutine, Buck2 SoftStartRoutine y BoostSoftStartRoutine desde el programa principal

Esta función lo que hace es pasar la referencia del valor inicial introducido en las funciones VoltageLoop, a su valor final mediante una rampa de subida que dura 50ms. Para ello se llama desde aquí a la función Delay_ms(1), está habilita las interrupciones provocadas por el timer y precarga el mismo para que las interrupciones se den cada 1 ms. De vuelta en la función SoftStartRoutine, se actualiza el valor de referencia con incrementos 50 veces más pequeños que la referencia final y se ubican en la posición de memoria correspondiente de la variable tPID (ControlReference). Si este valor de referencia no ha llegado al valor final, se repite el proceso en un bucle hasta llegar a la referencia final (50 pasos).

6.2.2.4.9 Delay_ms

Esta es la función que se explicó antes. Esta recibe de los SoftStartRoutine un número, en este caso 1, que será en número de desbordes del timer que el programador desea antes de parar el timer y sus interrupciones. Para ello, se precarga el timer para que cada desborde del mismo sea cada 1 ms, y además se habilita su interrupción. De este modo, se tiene un desborde del timer en 1 ms y después se desactiva el timer hasta la próxima llamada desde la SoftStartRoutine que está haciendo la rampa (50 veces). Si Delay_ms(Arg), recibiese 4, los desbordes del timer serían de 1 ms igualmente, pero hasta deshabilitarse el timer, deberían haber 4 desbordes.

Con esto se consigue tener retardos de 1 ms que son llamados hasta 50 veces por el SoftStartRoutine. De este modo se construye una rampa de subida en la referencia a seguir hasta llegar a su nivel final que se quedará fijo.

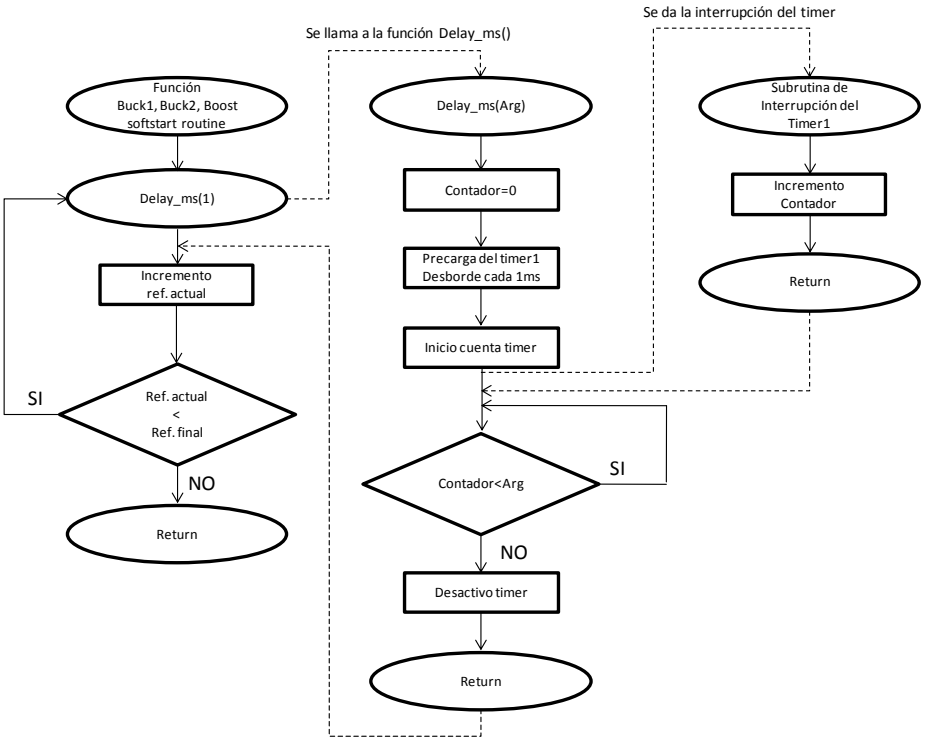


Figura 6-12 Flujograma de SoftStartRoutine, más retardo, más interrupción del timer1

6.2.3 Isr.c

En este fichero se han escrito las instrucciones de las subrutinas de atención a interrupción del timer 1 (retardo de 1ms) y de la interrupción externa 1 (Botón placa *starter board*). Estas interrupciones están definidas en C, y para que el programa dsPIC las identifique como tales, deben enunciarse unívocamente como:

_T1Interrupt() para la interrupción del timer 1

_INT1Interrupt() para la interrupción externa 1

Si no se hiciera así, el *Program Counter* no encontraría su posición en el *Interrupt Vector Table* de la memoria de programa, y no podría llegar a las subrutinas para poder ejecutar el dsPIC sus instrucciones.

La interrupción del timer, lo único que hace es incrementar un contador, que se comparará con el argumento de Delay_ms(Arg) para después resetear el flag de esta misma interrupción, que se activa por hardware y se limpia por software precisamente aquí con una instrucción. Con esto si el argumento es 1, el contador se incrementara sólo 1 vez y se desactivará el timer, con lo que se tiene 1 ms en el Delay_ms(Arg)

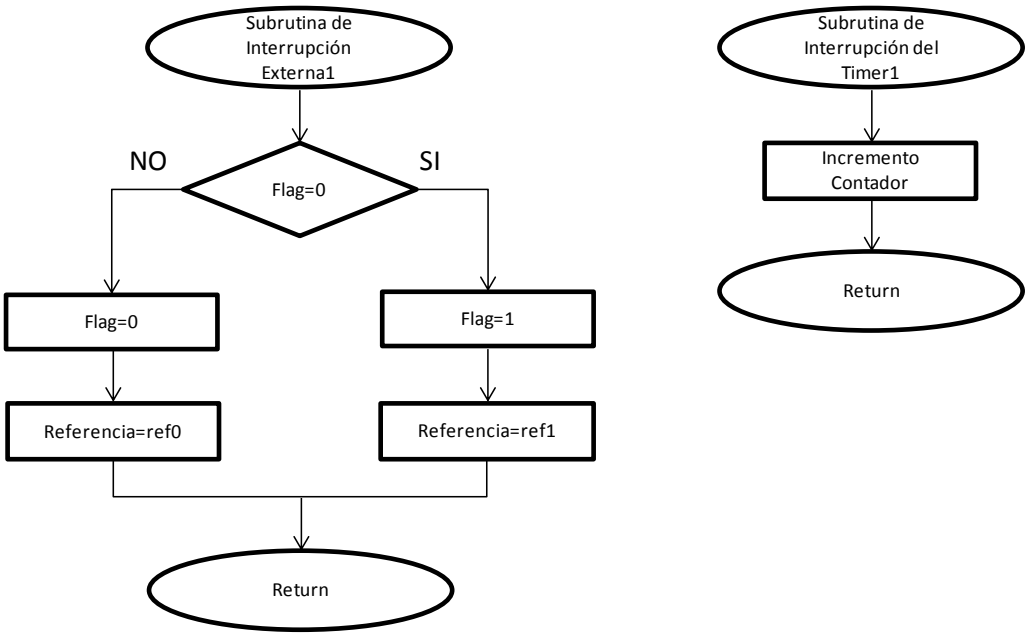


Figura 6-13 Flujograma de las subrutinas de atención a interrupción del timer1 y externa1

La subrutina de la interrupción externa se encarga de cambiar la referencia a seguir por el control. Para ello fue necesario declarar variables externas (ya que se usan aquí y donde fueron declaras) para referencia 1 y 2, un flag, que se usará para saber en qué referencia se está actualmente para así poder fijar hacía cual se va ahora, y la variable tPID del convertidor, que se usa para poder acceder a su campo referencia y modificarlo desde aquí. Para ello se usa una sentencia condicional en función del estado del flag, y se va por un lado u otro, para cambiar la referencia al valor que no estaba anteriormente. Los dos caminos tienen las mismas instrucciones por lo que en cualquier caso se tarda lo mismo en ejecutar un camino u otro. Tras hacer el cambio en la posición de memoria para la referencia de la variable tPID, es decir cuando se hace efectivo el cambio de referencia, se resetea el flag de la interrupción.

6.2.4 Isr_asm.s

Esta es la subrutina de atención a la interrupción del ADC. Esta escrita en ensamblador ya que así se tiene un mejor control del flujo e instrucciones del programa, ya que si se usan compiladores para C, ese control lo pierde el programador. Es aquí donde se empieza el proceso de cálculo del valor de control, y por ello el control sobre el programa debe ser alto para garantizar que estas instrucciones y con ello todo el algoritmo de control, duran siempre el mismo tiempo. Todas estas instrucciones y cálculos, incluso en el caso más desfavorable de reguladores tipo3, tardan menos de un periodo de PWM, por lo que es posible hacer los cálculos una vez por periodo. Esto es así siempre y cuando la frecuencia de conmutación no cambie. Experimentalmente se ha comprobado que conmutando a 300KHz da tiempo a realizar todo el proceso de cálculo.

En esta función se declaran unos valores máximos y mínimos para los valores de control que se van a cambiar en este proceso, asegurando que la modulación no satura. El flujo secuencial del programa se inicia poniendo a 1, un bit que nos servirá para poder medir cuanto tiempo se tarda en hacer todo el cálculo del control. Esto sólo se ha hecho en los casos en los que se trabaja con un solo convertidor, ya que para poder luego medir ese tiempo se necesitó un punto accesible de la placa Buck/Boost, siendo en estos casos el mismo pin del PWM1xL,

con lo que se inutiliza el reductor 1. De todos modos esto sólo se ha hecho así en los casos de tener que medir cosas concretas usando sólo uno de los convertidores, pero en realidad, para el funcionamiento del convertidor esto no hace falta, es sólo para medir el tiempo que tarda en actualizar la señal de control.

Aparte de este detalle, lo primero que se hace es hacer que un registro de trabajo (W0), tome el valor de la dirección donde empieza la variable tPID del convertidor por el cual se ha dado esta interrupción, a través del evento de su PWM. Así, el programa podrá moverse en estas zonas de memoria a partir del valor guardado en este registro de trabajo. Después se guarda en otro registro de trabajo el valor guardado en el ADCBUF correspondiente. Es en estos buffers, donde el ADC guarda el valor leído. En la práctica sólo se han usado el ADCBUF3 y el ADCBUF1 que son los de tensión, el buffer 3 para reductor 2 ó elevador y el 1 para el reductor 1, ya que sólo se han hecho controles en lazo simple de tensión. Este valor leído en el buffer del ADC tiene 10 bits de resolución, y como los valores que maneja el dsPIC son de 16bits, este valor de lectura se escala mediante un corrimiento de 5 posiciones hacia la izquierda del registro, con lo que queda en la parte más significativa del registro de trabajo de destino W1 (esto introduce una ganancia en el lazo que se detallará más en el apartado 7.1.2.2.2, y que se introduce en SmartCtrl mediante el sensor, 5.1.2). Tras esto, el valor medido y escalado se guarda en la posición de memoria de tPID correspondiente (MeasuredOutput), gracias a que mediante W0 lo tenemos localizado en la memoria de datos.

Tras esto se llama a la función PID, de la cual se hablará a continuación (6.2.5).

Tras la vuelta desde la función PID, se tiene ya en memoria de datos el nuevo valor de control que ha salido del algoritmo de la función PID, ya sea tipo3, 2 ó PI. Se extrae de ahí y se escala para hacer esa señal de control más significativa. Este escalado introduce de nuevo otra ganancia adicional en el regulador que se comentó ya en el capitulo 5.1.2, que en SmartCtrl se introduce mediante el sensor y que se detallará más en el apartado 7.1.2.2.4, ya que este escalado es mayor o menor en función del regulador implementado, por lo que se verá caso por caso en el capítulo de medidas experimentales. Después de este escalado se comprueba si está dentro de los límites fijados. Si es así, se introduce el nuevo valor de control en el PDC, con lo que este, se modula en una señal PWM con su ancho correspondiente. Si el valor calculado estuviese por encima de límite superior, se pondría en PDC ese máximo, y si fuese menor que el límite inferior, se metería este mínimo. Con esto se consigue que el PWM nunca esté saturado, ya que se evita al tener estos limitadores, que además son fácilmente configurables y modificables. Finalmente se resetean los flag que están a 1 por hardware, que son el de la interrupción y el de lectura de ADC completada que se han dado ambos previamente. En caso de estar midiendo el tiempo que se tarda en hacer el cómputo del valor de control, es en este momento, donde se resetea el bit usado para medir este tiempo, y que cuya pata para medirlo corresponde al PWM1xL, que en estas condiciones no funciona como tal. De forma esquemática reducida el diagrama de flujo de esta subrutina es la siguiente.

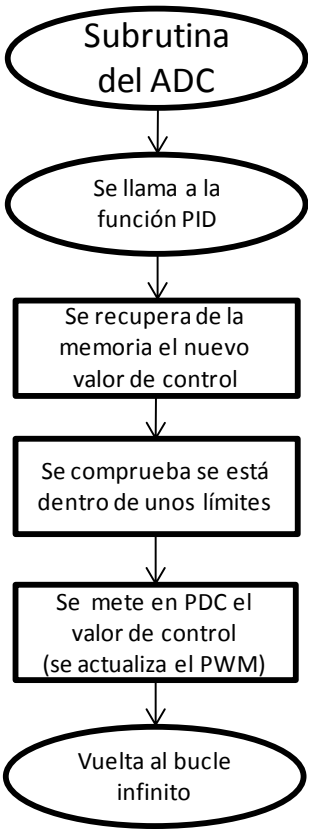


Figura 6-14 Subrutina de atención a la interrupción del ADC

6.2.5 PID.s

En un principio se describirá el código del algoritmo tipo3, por ser el más complejo y después se comentarán los cambios para los más sencillos. Esta función es llamada desde la interrupción del ADC y debido a que es parte del algoritmo de cálculo de control, se escribe en ensamblador para tener el máximo control del programa.

El algoritmo de control, hay que recordar que proviene de la FDT en discreto

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}}{a_0+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}} \qquad \textit{Ec. 6-5}$$

Donde en condiciones normales a_0=1

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0+b_1z^{-1}+b_2z^{-2}+b_3z^{-3}}{1+a_1z^{-1}+a_2z^{-2}+a_3z^{-3}} \qquad \textit{Ec. 6-6}$$

Con lo que nos queda una ecuación en diferencias tal que:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} + b_3 \cdot e_{n-3} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} - a_3 \cdot d_{n-3}$$

Ec. 6-7

Donde se recuerda que los coeficientes de la FDT son los que van desde b_0 a b_3 y de a_0 hasta a_3. Los valores de error son los que van desde e_n hasta e_{n-3} y los del ciclo de trabajo van de d_n a d_{n-3}, donde d_n será el nuevo valor de control que se va a calcular.

Finalmente queda que este es el cálculo que va a realizar el dsPIC, mediante la programación del algoritmo en ensamblador.

El flujograma simplificado es el siguiente:

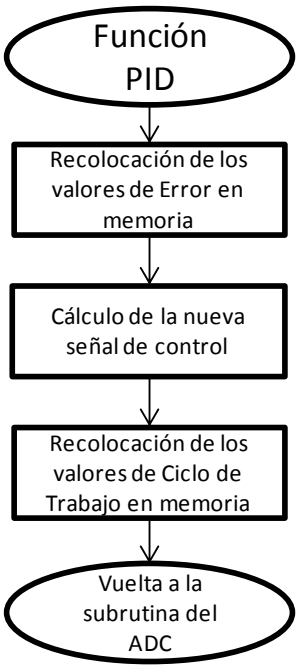


Figura 6-15 Flujograma de la función PID

Donde los valores a manejar en los cálculos son lo que se muestran en el ejemplo real de valores en la memoria de datos que se ve a continuación, estos han sido extraídos de un caso real de una variable de tipo tPID que se explicó en el capítulo 6.2.2.3

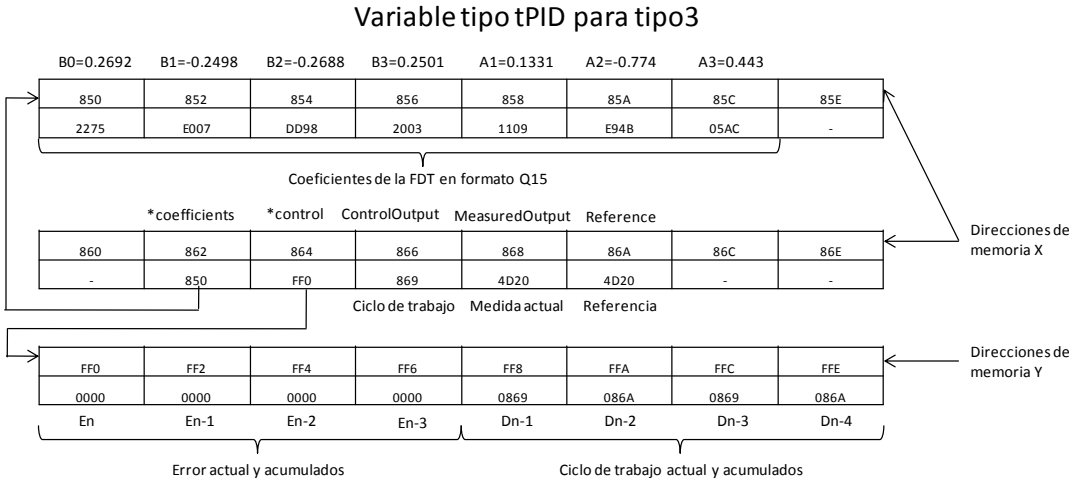


Figura 6-16 Memoria para una variable tPID en un regulador tipo3

6.2.5.1 Recolocación de valores de error

En un primer momento en este código PID.s, se recolocan los valores de error en sus nuevas posiciones dejando libre la primera.

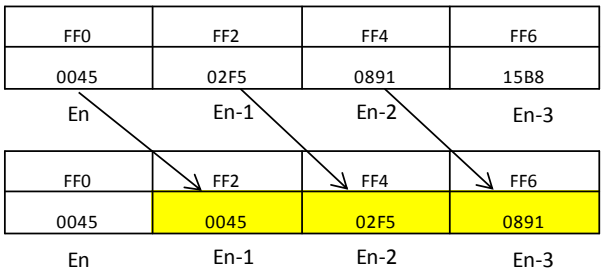


Figura 6-17 Recolocación en memoria de los valores de error

A continuación, esta primera posición se rellena con el nuevo valor de error, que sale del cálculo:

Error_n(E_n)=Referencia-medida actual

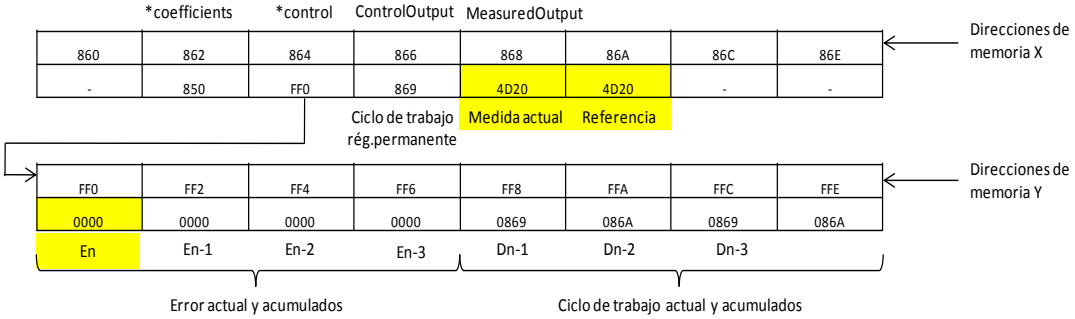


Figura 6-18 Error_n(E_n)=Referencia-medida actual=0000

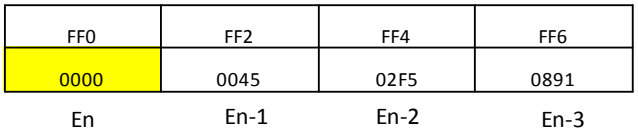


Figura 6-19 El último valor de error calculado va a En

Con lo que el vector de errores ya esta actualizado completamente para continuar con el cálculo del PID.

6.2.5.2 Cálculo del algoritmo de control

A continuación se realiza el cálculo del algoritmo. En este proceso tiene especial importancia la operación MAC, multiplicar y acumular. Es una operación de tipo DSP dónde se realiza la multiplicación de los datos ubicados donde apuntan dos registros de trabajo, uno a la memoria X (coeficientes) y otro a la memoria Y (Error y ciclo de trabajo anteriores). Posteriormente suma este valor a lo que hay en el acumulador, que en la primera multiplicación es 0 y en las sucesivas es el valor de la cuenta previa. Adicionalmente, esta instrucción guarda en unos registros de trabajo, las nuevas posiciones de memoria a donde apuntar en sucesivas cuentas (apuntando al siguiente coeficiente y al siguiente error-ciclo de

trabajo), estos son W8 y W10, que hacen esta operación mediante un post-incremento. Sin embargo los valores que son apuntados por estos, es decir los números útiles a multiplicar, los mete en W4 y W5 y una vez se meten ahí esos valores, es cuando se hace el post-incremento de w8 y w10, para apuntar a las siguientes posiciones.

Dicho esto, a continuación se muestran el flujograma y algunos gráficos explicativos.

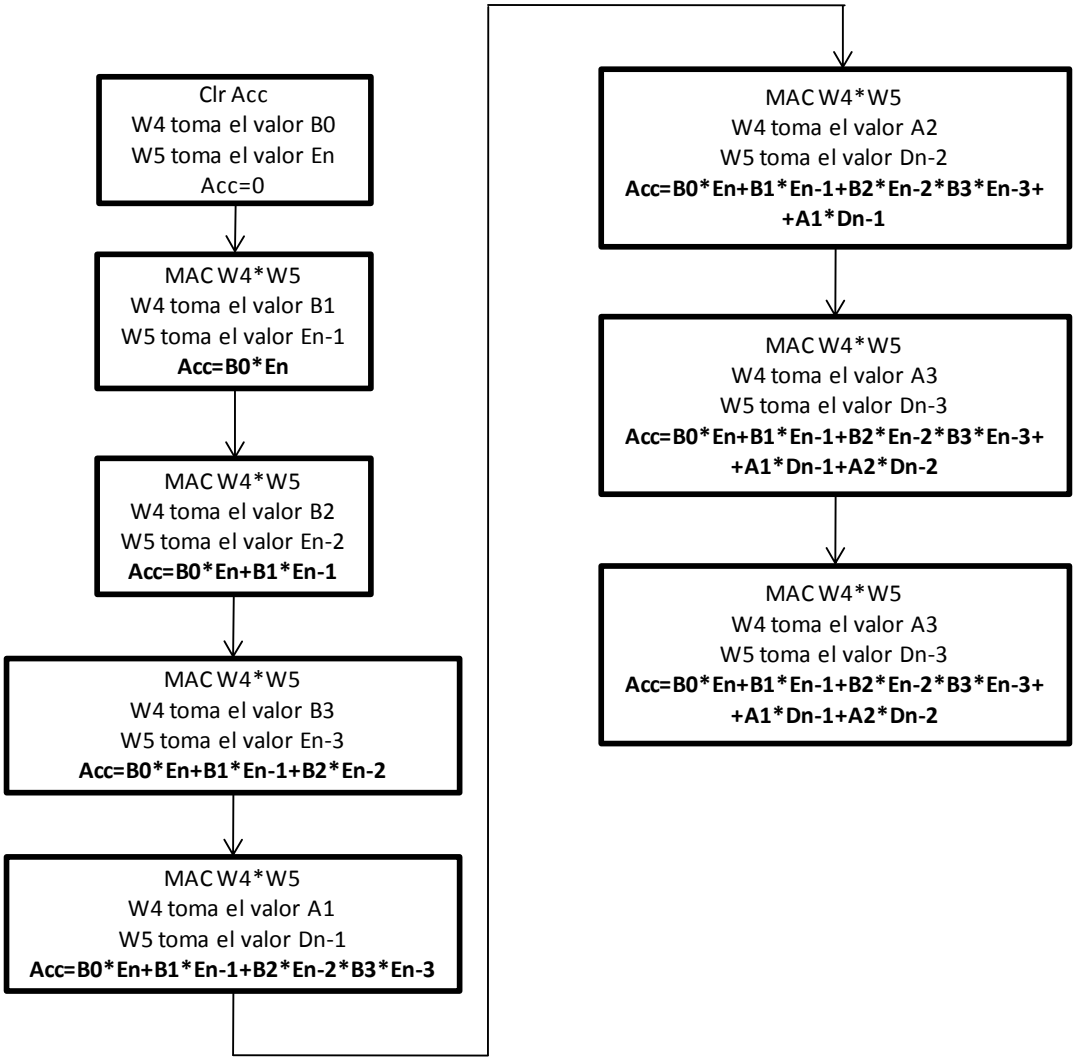


Figura 6-20 Flujograma del algoritmo de cálculo

Por lo tanto en este proceso, W4 y W5 van tomando los valores que hay en memoria de datos X e Y. Como se ve en la Figura 6-20, W8 y W10 apuntan a direcciones de memoria, pero son los valores que son apuntados por estos, los que quedan guardados en W4 y W5. En la figura el índice indica el número de operación, pero realmente sólo hay registros de trabajo W8, W10, W4 y W5, donde W8 y W10 recorren la memoria para extraer los datos a guardar en W4 y W5.

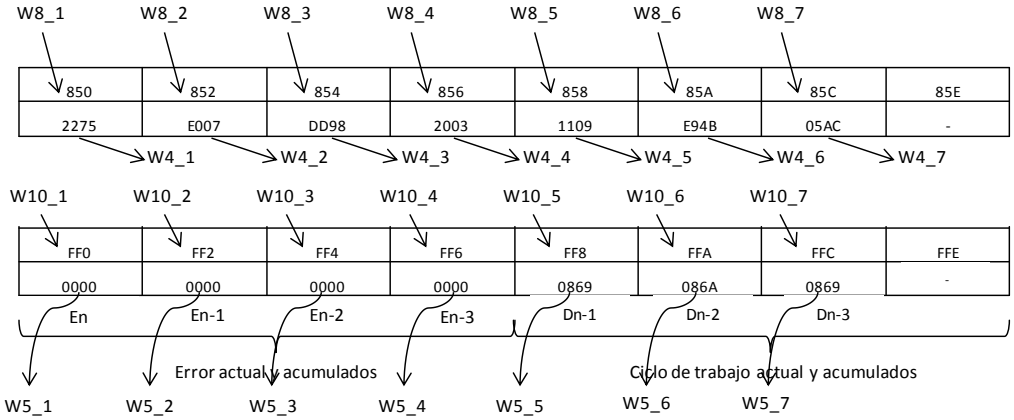


Figura 6-21 W4 recorre los coeficientes y W5 los errores y ciclos de trabajo

A continuación se muestra un ejemplo de las instrucciones en ensamblador por su especial interés en el diseño del regulador. Sin embargo las instrucciones mostradas a continuación sólo funcionarían en caso de que todos los coeficientes pudiesen ponerse en formato Q15 (todos entre 0 y menor que 1 en valor absoluto). Por lo tanto este ejemplo es explicativo, pero difícilmente válido en un caso real.

```
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]-=4, w5
mac     w4*w5, a,
```

Aquí se puede ver que en cada instrucción se multiplica, se mete el valor contenido en [W8] y [W10] en W4 y W5 y finalmente se hace un post-incremento de W8 y W10. En estos post-incrementos se ve que W10, que apunta al vector de error y ciclo de trabajo (memoria Y), al final se recoloca apuntando a D_n.

6.2.5.3 Recolocación de valores de ciclo de trabajo

Una vez se tiene en el acumulador el nuevo valor del los cálculos, este valor es redondeado y guardado en un registro de trabajo. Esto es así porque el acumulador es de 40 bits pero los registros de trabajo y posiciones de memoria son de 16, por lo tanto se redondea para tomar la parte más significativa del acumulador y meterla en un registro de 16 bits. Después se realiza un proceso de recolocación de los valores similar al del punto 6.2.5.1, pero ahora con los valores de ciclo de trabajo. Sin embargo hay un importante detalle que comentar. Al iniciar los cálculos para llegar al ciclo de trabajo nuevo (d_n), se parten de valores que ya de por sí, son del muestreo anterior, de ahí que en las figuras de las variables tPID, se parte de D_{n-1} hasta D_{n-3}. Es decir, al iniciarse los cálculos para obtener la nueva señal de control, los cálculos se realizan con muestras desde la anterior hasta otras previas, con lo que la más reciente que se tiene es D_{n-1}. Una vez se tiene el nuevo valor de control calculado, ya si se tiene

el D_n de este ciclo. Y por lo tanto para hacerle sitio en la memoria se desplazan las posiciones de todos, como se ve en la Figura 6-22.

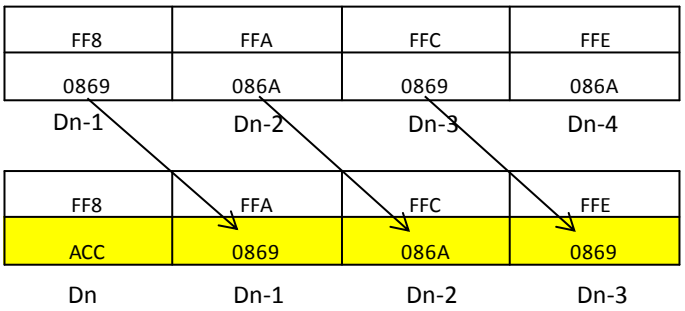


Figura 6-22 Recolocación en memoria de los valores de ciclo de trabajo

Por lo tanto se tiene que en la posición de D_n final, se coloca el nuevo valor calculado y redondeado que se tomó del acumulador. Esto se realiza mediante una instrucción ya que [W10] se dejó precisamente apuntando donde se va a guardar el nuevo valor D_n. Tras esto se vuelve a la función que llamo a esta, que es la de interrupción del ADC y es esta última, la que accede a la memoria de datos, en concreto a la posición FF0, donde esta contenido el nuevo valor de control (d_n) para introducirlo en el registro PDC y así ajustar el ciclo de trabajo a la nueva situación. Esta actualización de ciclo de trabajo como ya se ha comentado anteriormente, se hace efectiva en el próximo periodo de PWM.

6.2.5.4 Diferencias del PID.s en reguladores de menor orden

Por último se van a comentar las diferencias en función de tipo de control programado. En este aspecto hay que centrarse en cómo son las FDT en los distintos casos. Como se ha mostrado anteriormente para el tipo 3 se tiene la siguiente ecuación en diferencias:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} + b_3 \cdot e_{n-3} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} - a_3 \cdot d_{n-3}$$

Ec. 6-8

En este caso, anteriormente se ha mostrado el tamaño de los vectores de datos, tanto para almacenar los coeficientes, como para los errores y ciclos de trabajo en sus correspondientes muestras.

Para el caso del tipo 2, la FDT tiene la forma:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \textit{Ec. 6-9}$$

Donde la ecuación en diferencias queda como:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} \qquad \textit{Ec. 6-10}$$

Respecto al tipo3, se ve que tiene menos coeficientes por lo que las zonas de memoria a reservar en la variable tPID son menores.

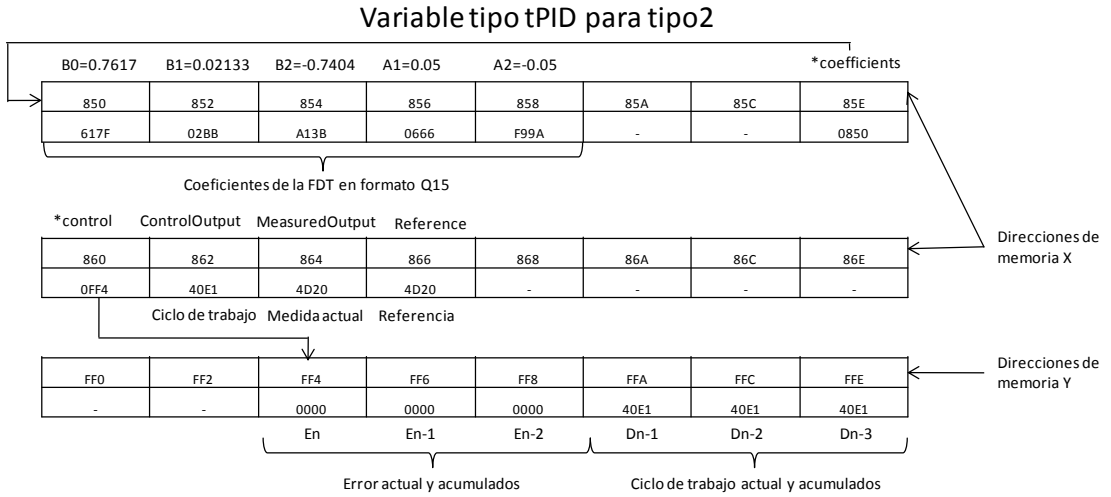


Figura 6-23 Memoria para una variable tPID en un regulador tipo2

En este ejemplo se muestra un caso real de las posiciones de memoria para un tipo 2. Se ve que los coeficientes son 5 y los errores y ciclos de trabajo son 3 y 3, frente a los 7 coeficientes y 4 y 4 errores y ciclos de trabajo del tipo 3.

Por último el tipo PI es el que tiene una FDT más sencilla:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} \qquad \textit{Ec. 6-11}$$

Cuya ecuación en diferencias es:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} - a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-12}$$

De nuevo esto afecta al número de espacios de memoria a usar para el cálculo de la señal de control.

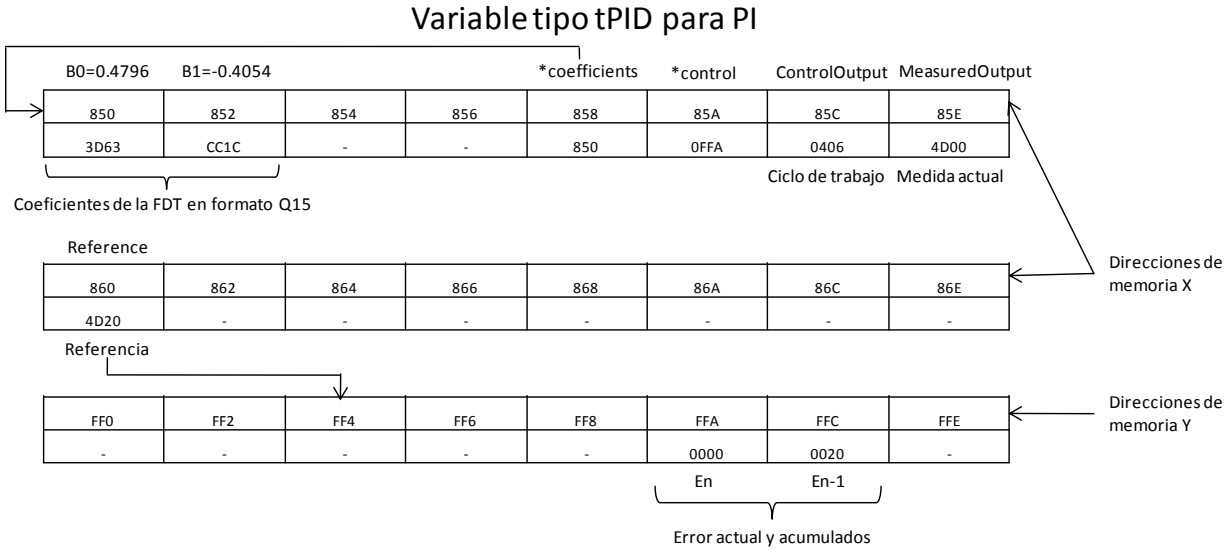


Figura 6-24 Memoria para una variable tPID en un regulador PI

Como vemos en este caso sólo se manejan dos coeficientes del numerador y dos muestras de error. Los coeficientes del denominador y las muestras de ciclo de trabajo (d_n) no son necesarios ya que de la ecuación:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-13}$$

El término

$$a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-14}$$

Se introduce mediante la suma al acumulador de d_{n-1}, al tener que a_1=1. Esto se hace así y no mediante la multiplicación a_1*d_{n-1}, para poder tener la función de integrador en el control que se explicará más detalladamente en el punto 6.2.5.5. Por lo tanto, en el algoritmo de control se multiplica:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} \qquad \textit{Ec. 6-15}$$

Y se le añade

$$a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-16}$$

Donde a_1=1, obteniéndose la ecuación en diferencias deseada:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-17}$$

Y que implementa en el dsPIC un control de tipo PI

6.2.5.5 **Implementación de la función de integrador**

Como se ha introducido anteriormente, en los tres tipos de control hay que tener la función de integrador. Para explicar esto se partirá de la siguiente ecuación genérica:

$$G(z^{-1}) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \cdots + a_n z^{-n}} \qquad \textit{Ec. 6-18}$$

Si observamos la FDT (Ec. 6-18) como dos polinomios, uno en el denominador y otro en el numerador, tenemos que el de abajo tiene la siguiente forma:

$$D(z^{-1}) = a_0 + a_1 z^{-1} + \cdots a_n z^{-n} \qquad \textit{Ec. 6-19}$$

Para nuestros casos, se necesita que el regulador realice la función de integrador. En continuo esto equivale a tener un polo en el origen (s=0), pero en discreto esto equivale a tener uno polo en z^{-1}=1, y en el polinomio de los polos, (el denominador de la FDT) si este tiene un polo en esa posición, se puede escribir de la siguiente forma:

$$D(z^{-1}) = (1 - z^{-1}) \cdot \left(\ldots z^{-(n+1)}\right) \qquad \textit{Ec. 6-20}$$

Donde

$$(1 - z^{-1}) \qquad \textit{Ec. 6-21}$$

Nos garantiza ese polo en z^{-1}=1 que haría la función de integrador del regulador.

Para que un polinomio pueda ponerse de la forma que se ha descrito antes, se debe cumplir una condición. Cuando evaluamos el polinomio con $z^{-1=}$1, el valor del sumatorio de las raíces del polinomio debe valer 0.

Es decir si tenemos

$$D(z^{-1}) = a_0 + a_1 z^{-1} + \cdots a_n z^{-n} \qquad \textit{Ec. 6-22}$$

Donde

$$z^{-1} = 1 \qquad \textit{Ec. 6-23}$$

Nos queda de la siguiente manera

$$D(1) = a_0 + a_1 + \cdots a_n = 0 \qquad \textit{Ec. 6-24}$$

Por eso se dice que es necesario en nuestro caso que el sumatorio de los coeficientes del denominador sea 0, ya que esta es la condición que garantiza que nuestro regulador introduce un polo en z^{-1}=1 (integrador).

Por lo tanto es imprescindible que en la FDT:

$$\sum a_n = 0 \qquad \textit{Ec. 6-25}$$

Este problema tiene una especial implicación a la hora de implementar el control de tipo PI. Recordemos que su FDT es:

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1}} \qquad \textit{Ec. 6-26}$$

Y la ecuación en diferencias para la parte del denominador es:

$$a_0 d_n = \cdots - a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-27}$$

Si a_0=1, para que se cumpla

$$D(1) = a_0 + a_1 = 0 \qquad \textit{Ec. 6-28}$$

Es necesario que a_1=-1

Pero en Q15 este número es imposible implementarlo, ya que solo puede manejar números discretos de 0 hasta el más próximo a 1, con 15 bits de resolución, pero sin llegar a 1. Es decir, que en valor absoluto el número más alto que se puede manejar en Q15 es:

$$2^{-1} + 2^{-2} + 2^{-3} + \cdots + 2^{-15} = 0.999969482421875 \qquad \textit{Ec. 6-29}$$

Por lo que no se garantiza la función de integración y sólo es posible aproximarse, con los consiguientes problemas en la ganancia del lazo a bajas frecuencias, que al no tener integrador, no tienen por qué ser suficientemente alta.

Como se explicó antes para implementar el PI, se necesitaba la siguiente ecuación en diferencias:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-30}$$

Donde a_1, debe ser igual que 1 para que se cumpla la condición de integrador. Para ello se calculan los productos

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} \qquad \textit{Ec. 6-31}$$

Y se le añade

$$d_{n-1}$$

Obteniéndose la ecuación en diferencias deseada:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + a_1 \cdot d_{n-1} \qquad \textit{Ec. 6-32}$$

Donde así a_1=1, que cumple la condición de integrador.

6.2.5.6 **Cálculos con números mayores de 1**

Como se ha ido explicando anteriormente, los cálculos que realiza el dsPIC, no pueden tener números mayores o iguales a 1. Sin embargo es posible que la FDT de la forma genérica siguiente:

$$G(z^{-1}) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{b_0+b_1z^{-1}+\cdots+b_nz^{-n}}{a_0+a_1z^{-1}+\cdots+a_nz^{-n}} \qquad \textit{Ec. 6-33}$$

Tenga números cuyo valor este fuera del formato Q15. Hay que recordar que en las operaciones de multiplicación con MAC comentadas anteriormente, sólo se realizaban con la parte fraccionaria de estos números. De tal modo que al introducir los coeficientes de la FDT en el programa, no se meten exactamente estos, ya que pueden estar fuera del formato Q15, sino sólo su parte fraccional. Por lo tanto para finalmente poder construir un algoritmo que calcule una ecuación en diferencias del tipo:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \ldots + b_m \cdot e_{n-m} - a_1 \cdot d_{n-1} - \cdots - a_m \cdot d_{n-m}$$

Ec. 6-34

Donde los coeficientes b_0 hasta b_m y a_1 hasta a_m pueden ser 1 ó mayores, es necesario hacer algún ajuste. En el caso de este proyecto, este problema se ha resuelto de la siguiente manera.

Supóngase un número mayor que 1 con parte fraccionaria que ha sido obtenido tras el proceso de diseño del regulador explicado en el capítulo 5. De él, se almacena su parte fraccionaria, que es tratable por el dsPIC en Q15, tal y como se explicó anteriormente en el fichero init.c y que irá a su posición donde indica la variable tPID. La parte entera, será tratada por el programador que hará el ajuste necesario en el algoritmo descrito en ensamblador del fichero PID.s. Por lo tanto si se tiene un número con parte decimal y entera, estas serán separadas y tratadas de una forma particular para conseguir que por ejemplo:

2.1452

Pase a ser

2+0.1452

Como ejemplo explicativo véase el siguiente:

Supóngase que se tiene que la FDT de un regulador de tipo 2 tiene los siguientes coeficientes:

$$G(z) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{2.156+0.254z^{-1}+1.784z^{-2}}{1+0.651z^{-1}+1.654z^{-2}} \qquad \textit{Ec. 6-35}$$

Sobre los coeficientes que están entre 0 y 1, no hay que hacer nada excepto introducirlos en el programa a través del fichero init.c en sus posiciones correspondientes, es decir:

B_1=0.254

A_1=0.651

Y para los que no están en formato Q15, se introduce en init.c sólo la parte fraccionaria, es decir:

$B_{0=}$0.156

B_2=0.784

A_2=0.654

Una vez se tienen estos coeficientes en su zona de memoria es necesario hacer ajustes en el algoritmo para conseguir una ecuación como:

$$d_n = b_0 \cdot e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} \qquad \textit{Ec. 6-36}$$

Donde el programador que introduce los coeficientes, sabe que:

B_0=2+0.156

B_2=1+0.784

A_2=1+0.654

Y por lo tanto para hacer bien el cálculo con los coeficientes que hay programados, se necesita que por ejemplo para el caso de b_0:

$$b_{0\,real} \cdot e_n = b_0 \cdot e_n + e_n + e_n \qquad \textit{Ec. 6-37}$$

Es decir

$$2.156 \cdot e_n = 0.156 \cdot e_n + e_n + e_n \qquad \textit{Ec. 6-38}$$

Lo que para el ejemplo general del tipo 2, queda

$$d_n = b_0 \cdot e_n + e_n + e_n + b_1 \cdot e_{n-1} + b_2 \cdot e_{n-2} + e_{n-2} - a_1 \cdot d_{n-1} - a_2 \cdot d_{n-2} - d_{n-2}$$

Ec. 6-39

Que con los números del ejemplo

$$d_n = 0.156 \cdot e_n + e_n + e_n + 0.254 \cdot e_{n-1} + 0.784 \cdot e_{n-2} + e_{n-2} - 0.651 \cdot d_{n-1} - 0.654 \cdot d_{n-2} - d_{n-2}$$

Ec. 6-40

Una vez se ha explicado esto, se comentará brevemente las instrucciones en ensamblador que hay que intercalar con los MAC. En este caso se trataría de la instrucción LAC, que toma el valor de un registro de trabajo, donde tenemos los valores de error (e) o de ciclo de trabajo (d), y lo mete en el acumulador libre, no en el que esta toda la cuenta de d_n. Posteriormente se hacen instrucciones ADD, que suman un acumulador al otro, o instrucciones SUB, que restan los acumuladores. De estas instrucciones se meterán tantas como sea necesario para meter los suficientes enteros que hacen falta. Lógicamente siendo ADD, si en la ecuación en diferencias iban sumando o SUB en caso de ir restando. Estas sumas o restas se realizan directamente en el acumulador donde se están haciendo todos los cálculos y por lo tanto no hay que añadir ninguna otra instrucción. De este modo a efectos prácticos el flujo del algoritmo de cálculo del regulador se corresponde con el que hay en la Figura 6-20.

Como ejemplo, se muestra la secuencia de instrucciones siguiente, donde se ve que dos de los coeficientes de la FDT son mayores que 1 pero menores que 2, ya que por lo tanto sólo hay un add y un sub. En este ejemplo se ve que una de las multiplicaciones va sumando y otra restando en la función de transferencia.

```
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
```

```
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
lac     w5,     b
add     a
mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
lac     w5,     b
sub     a
mac     w4*w5, a, [w8]+=2, w4, [w10]-=4, w5
mac     w4*w5, a,
```

Un ejemplo de FDT que funcionaría realmente con este algoritmo de cálculo sería:

$$G(z) = \frac{Y(z^{-1})}{X(z^{-1})} = \frac{0.8618 - 0.8096z^{-1} - 0.861z^{-2} + 0.8104z^{-3}}{1 - 1.835z^{-1} + 1.01z^{-2} - 0.1744z^{-3}} \qquad \textit{Ec. 6-41}$$

Donde a_1 es menor que -1 y a_2 es mayor que 1.

7 Medida experimental de los lazos de control digitales

En este capítulo se describen los procedimientos de medida que se han utilizado en el proyecto. El montaje de un prototipo y la obtención de medidas experimentales es un paso fundamental en la validación del proceso de diseño seguido.

Las medidas experimentales que se han realizado en este proyecto se pueden separar en dos grupos en función de la instrumentación usada o del objetivo de la medida:

- Medidas con osciloscopio de respuesta temporal. Son las primeras en hacerse para ver que el lazo funcionaba
- Medidas de respuesta en frecuencia mediante analizadores de ganancia y fase. Se comprueba que la respuesta en frecuencia medida experimentalmente se asemeja a la calculada en el diseño del control con SmartCtrl. Estas medidas son las que sirven de validación de la herramienta de diseño. Obtención de la respuesta temporal

En primer lugar se han realizado simples medidas mediante un osciloscopio para ver si el control regulaba una vez había sido programado el dsPIC, mediante la variación de la tensión de entrada o escalones en la referencia. Este tipo de medidas resultaron muy útiles en los primeros pasos del proyecto cuando el lazo de control no funcionaba por distintos problemas (en principio, lo más habitual era el no funcionamiento por estar mal programado el control). En esta fase fue empleado el montaje que se ve en la Figura 4-14, donde se implementaban distintos programas hasta que finalmente, se han conseguido los tres algoritmos de control distintos que han sido recogidos en los resultados experimentales de este proyecto, el PI, tipo 2 y tipo 3.

La primera prueba que se ha hecho es simplemente variar la tensión de entrada a los convertidores y medir la tensión de salida y los pulsos del PWMxH. Al variar la tensión de entrada la salida permanecía fija en el valor prefijado y la anchura del PWM variaba para conseguir que la salida quedase fija. Este ensayo era inmediato para ver si lo que se había implementado funcionaba. Era la primera prueba que se hacía a cada control, ya que es rápida y aporta información de si el lazo está o no correctamente cerrado. Pese a ser una prueba muy sencilla y poco exigente, ha resultado muy útil por su rapidez en aportar esta primera información sobre el lazo. Además de estas medidas también se han hecho medidas de respuesta temporal, como pueden ser escalones en la referencia o las medidas de tiempo de cálculo del algoritmo de control, sin embargo estas no han sido incluidas en la memoria ya que son poco relevantes frente a las medidas de respuesta en frecuencia, sobre las que se ha profundizado bastante para comprobar que el proceso de diseño de reguladores definido en este proyecto es válido experimentalmente.

Una vez se obtenían estas informaciones de respuesta temporal, se han hecho las medidas de respuesta en frecuencia de planta, regulador y lazo.

7.1 Obtención de la respuesta frecuencial

En esta parte de las medidas experimentales es donde más hincapié se ha hecho, ya que se ha buscado obtener la máxima fidelidad entre lo implementado y lo que se esperaba teóricamente obtenido de SmartCtrl y MatLab. Como se introdujo en el apartado 2.3.2, el objetivo de este tipo de medidas es conocer la dinámica del componente medido tanto para

variaciones rápidas como lentas. Mediante estas medidas sobre sistemas físicos, se obtienen las respuestas en frecuencia experimentalmente, lo que permite comprobar la estabilidad del sistema y verificar los modelos matemáticos que se hayan utilizado en el diseño. Gracias a esta información experimental, se puede saber si el lazo es estable, que grado de estabilidad tiene, cómo es posible mejorarla y que márgenes de estabilidad se tiene en el sistema.

Por otro lado, teóricamente partiendo de estas medidas en frecuencia, se puede obtener también información de la respuesta transitoria. Los márgenes de fase y ganancia dan información del amortiguamiento del sistema y la frecuencia de cruce, la de resonancia y el ancho de banda dan información de la rapidez de respuesta transitoria. [2]

Como se explicó en el capítulo 5 para el diseño del control, se parte de la respuesta frecuencial de la planta o sistema a controlar. Por lo tanto en el proceso de diseño, lo primero que se ha hecho es obtener esta respuesta frecuencial del reductor o del elevador y una vez se tiene, se realiza todo el proceso de diseño. Con el nuevo control implementado, se realizan medidas de regulador, lazo y de nuevo la planta para comprobar si lo implementado y lo que se diseño mediante SmartCtrl coinciden. Por lo tanto esta obtención de la respuesta frecuencial, ha sido parte sustancial de este proyecto por su importancia tanto en la obtención de resultados, como en los primeros pasos del diseño.

7.1.1 Medidas de respuesta frecuencial

En la Figura 7-1 se muestra un esquema simplificado y genérico de la medida de un bloque, donde mediante la medida de referencia y test se obtiene la respuesta frecuencial experimentalmente de un bloque a priori desconocido.

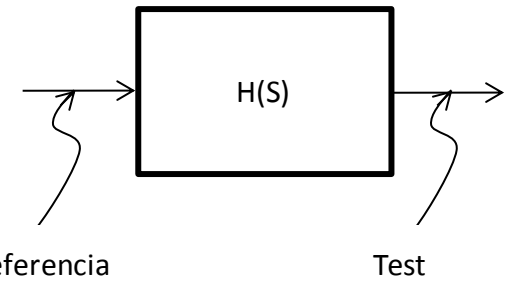


Figura 7-1 Esquema genérico de medida experimental de un bloque

La medida de la respuesta en frecuencia de un sistema funcionando en lazo cerrado consiste en la introducción de una perturbación sinusoidal cuya frecuencia va cambiando a lo largo del proceso de medida. Esta señal se debe introducir en el sistema de manera que siga funcionando en lazo cerrado (Figura 7-2). Sin embargo, para poder sumar esta señal es necesario poder abrir el lazo de control e inyectar esta señal de frecuencia variable y flotante en el lazo. En nuestro caso, esta ruptura del lazo se hace en la placa de los convertidores en los jumpers del reductor 2 ó del elevador. Por lo tanto para introducir esta perturbación no ha sido necesario abrir el lazo de forma forzada sobre la placa *Buck*/*Boost*. A través de estos jumpers se ha introducido una perturbación generada con los analizadores de ganancia y fase del laboratorio, y mediante sus sondas se mide como varía esta perturbación entre un punto tomado como referencia y otro tomado como test, donde la sonda referencia se conecta a la entrada del bloque a medir, y la sonda test, se conecta a la salida del mismo. A partir de la comparación entre ambas medidas, se obtiene la ganancia (variación de amplitud) y la fase (variación de la fase), entre el punto de referencia y el de test para una frecuencia,

repitiéndose este proceso para todo un barrido de frecuencias desde 10Hz hasta 400kHz, normalmente en este proyecto, aunque este rango ha podido ser variado en algunas medidas de pruebas. Para la inyección de la perturbación se ha usado el sistema que se propone en [3]. Este consiste en la introducción de la perturbación mediante el uso de un transformador, a cuyo primario se conecta el oscilador del analizador de frecuencia y fase que genera la perturbación y en el secundario una resistencia de 100Ω ó menos en serie con el lazo que ha sido previamente abierto.

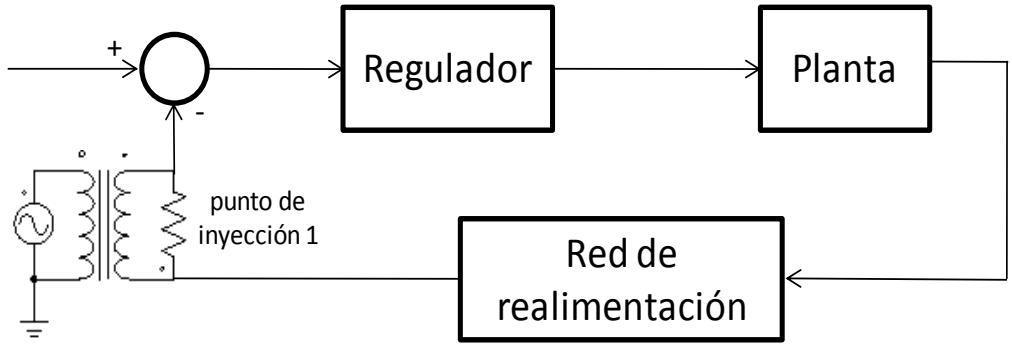


Figura 7-2 Esquema general del lazo a medir

En el caso de este proyecto la perturbación se ha inyectado en ese punto precisamente porque se dispone de una forma sencilla de abrir el lazo, aunque es posible introducirla en otros puntos. Sin embargo es en este punto donde mejores resultados se obtienen a altas frecuencias, lo que en cierto modo nos interesa para poder llegar a medir lazos rápidos, y conocer los efectos del retardo digital con mayor precisión ya que han resultado ser un factor crítico en el diseño de estos reguladores. Sin embargo en bajas frecuencias resulta algo peor por lo que por debajo de 100Hz la medida puede quedar desvirtuada. [3]

Otro factor importante para realizar esta medida correctamente es el de la selección de la amplitud de la perturbación introducida. Para seleccionar esta amplitud hay que tener en cuenta que normalmente cuanto mayor es la medida mejor salen los datos obtenidos, con más repetitividad entre una medida y otra. Sin embargo si se introduce una perturbación muy amplia, puede dar lugar a no linealidades dejando de considerarse esta como pequeña señal al alejarse mucho del punto de trabajo y en un caso extremo, incluso provocar daños en algún componente del lazo al correr por él la tensión flotante inyectada con una amplitud excesiva. Por lo tanto es importante usar la amplitud apropiada que aporte buena repetitividad en las medidas pero pudiendo considerarse esta como pequeña señal.

7.1.2 Componentes del lazo y su medida.

Los bloques sobre los que se han realizado medidas en este proyecto han sido la planta, el regulador y el lazo completo, aunque en algunos casos se ha caracterizado también el sensor (red de realimentación), pero no en todos lazos de los resultados finales. En cualquier caso, para la medida de cada bloque existen particularidades que van a ser detalladas a continuación.

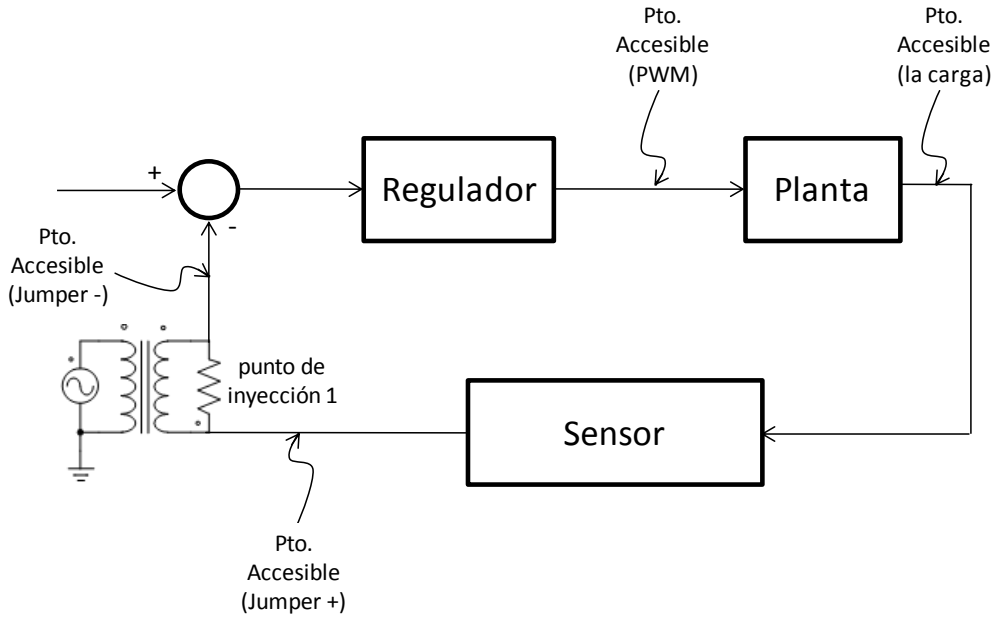


Figura 7-3 Esquema general del lazo con sus puntos accesibles

Antes de detallar los distintos bloques hay que mencionar que las medidas están limitadas por los puntos accesibles del lazo, que son los que se ven la Figura 7-3. Este hecho hace que, por ejemplo, dentro del regulador haya más bloques: además del propio algoritmo de control existen distintas ganancias adicionales. Estas ganancias no se han podido separar en la medida del algoritmo principal porque a la hora de medir, sólo existen los puntos accesibles que se ven en la figura. Otro ejemplo de esta limitación es que no se puede medir directamente la señal de ciclo de trabajo, sino que se tiene que medir sobre los pulsos de control generados por el modulador PWM.

7.1.2.1 La planta

Como se explicó en el apartado 5.1.1, la planta ha sido medida previamente para poder importarla a SmartCtrl. Una vez medida e importada se ha hecho todo el proceso de diseño con lo que la medida de la planta se ha hecho en un primer momento para importarla a SmartCtrl, y posteriormente se han medido en todos los casos experimentales desarrollados en este proyecto. Esta pequeña diferencia con los otros bloques de las medidas de los resultados (regulador y lazo) no afecta en absoluto a la forma de medir, siendo en ambos casos la que se describirá a continuación, donde el esquemático siguiente muestra la colocación de las sondas test y referencia del analizador para medir la planta.

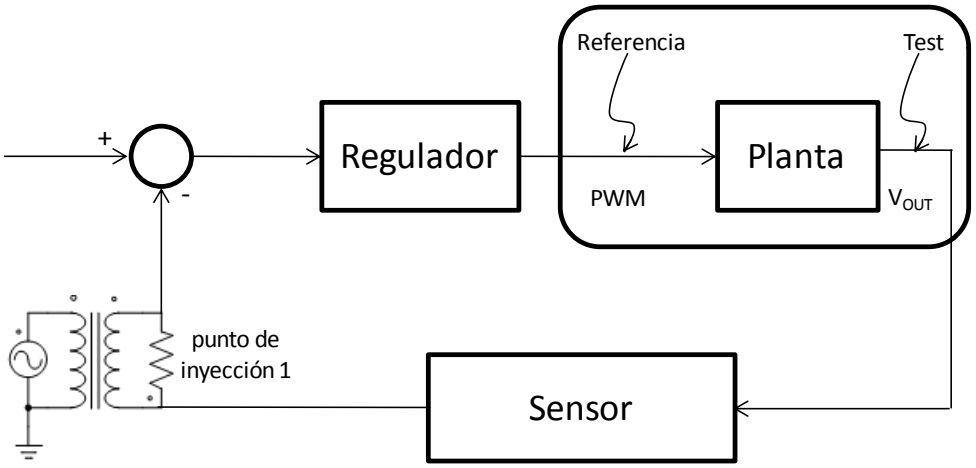


Figura 7-4 Esquema de medida de la planta.

Como se ve en el esquema de la Figura 7-4, la sonda de referencia queda entre la salida del regulador (salida del dsPIC) y la entrada de la planta. La sonda de test se conecta a la carga (tensión de salida), que es la salida de la planta. Como puede verse, la sonda de referencia va directamente sobre una señal en forma de PWM, ya que el ciclo de trabajo d, no es accesible porque la salida del dsPIC son directamente los pulsos PWM donde está contenida la señal de control d. Esta característica requiere de una pequeña compensación de la medida.

Para el caso de la medida de la planta (G_{vd}), se quiere medir la tensión de salida con respecto al ciclo de trabajo d. Sin embargo en este caso, d no es un número entre 0 y 1, sino una señal PWM cuyos valores van entre 0 y 3,3V. De tal modo que si d fuese máximo, es decir 1, en la señal PWM se tendría 3,3V y no 1V. Por lo tanto si para caracterizar la planta se necesita comprar d con la salida, en estas condiciones se tiene que la entrada tiene por la modulación una ganancia de 3,3 que habrá que corregir. En la salida no es necesario hacer ningún ajuste, ya que para la medida de la planta (G_{vd}), sí se debe medir directamente la tensión de salida, y por lo tanto con la sonda en ese punto, no es necesario realizar ninguna compensación más. Finalmente si se mide un bloque cuya entrada es X y su es salida Y, como se han descrito anteriormente, sobre la medida que nos da el analizador se tiene que a la entrada:

$$X = 3{,}3 \cdot d \qquad \textit{Ec. 7-1}$$

Y a la salida

$$Y = V_{out} \qquad \textit{Ec. 7-2}$$

Con lo que finalmente la FDT de la salida (V_o) frente a la entrada (ciclo de trabajo), queda:

$$G_{vd} = \frac{Y}{X} = \frac{V_{out}}{3{,}3 \cdot d} = \frac{V_{out}}{d} \cdot \frac{1}{3{,}3} \qquad \textit{Ec. 7-3}$$

$$20 \cdot log\left(\frac{1}{3{,}3}\right) = -10{,}37dB \qquad \textit{Ec. 7-4}$$

Con lo que la medida estará atenuada por efecto de la modulación de d, en 10.37dB, que se deben añadir a los valores de la medida para que la planta se ajuste al caso real. En

todas las medidas experimentales del apartado 8, esta corrección ya se ha añadido por lo que los resultados de la planta que se muestren se ajustarán a la planta real.

7.1.2.2 El regulador

Cuando se considera el regulador desde el punto de vista de la medida hay que tener en cuenta que sólo son accesibles algunos puntos del dsPIC, que es quien realiza las funciones de regulador. Es decir, en estas medidas el bloque del regulador, incluye al regulador en sí mismo, pero también al modulador PWM y al ADC, que también introducen ganancias y retardos, e incluso otras ganancias que provoca el código programado en el dsPIC.

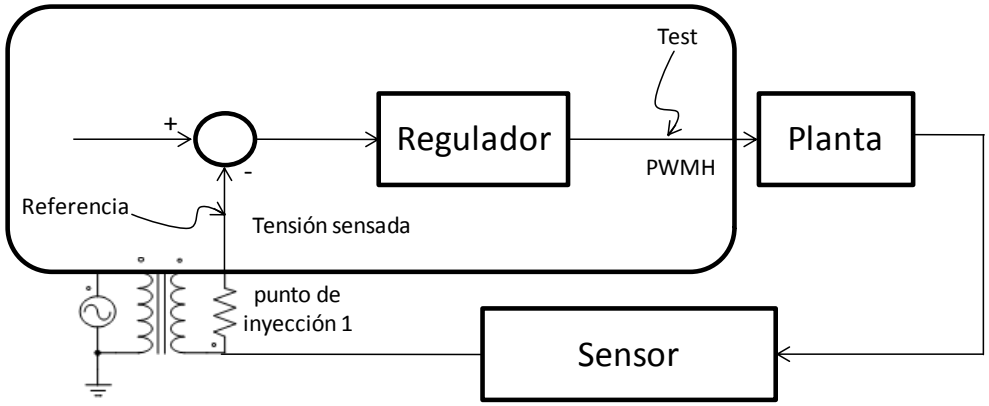


Figura 7-5 Esquema de medida del regulador

En la Figura 7-5 se ve el esquema general condicionado por los puntos accesibles para la medida del regulador, y en la Figura 7-6, el detalle del bloque del regulador donde se ven los bloques detallados que intervienen en el valor medido experimentalmente.

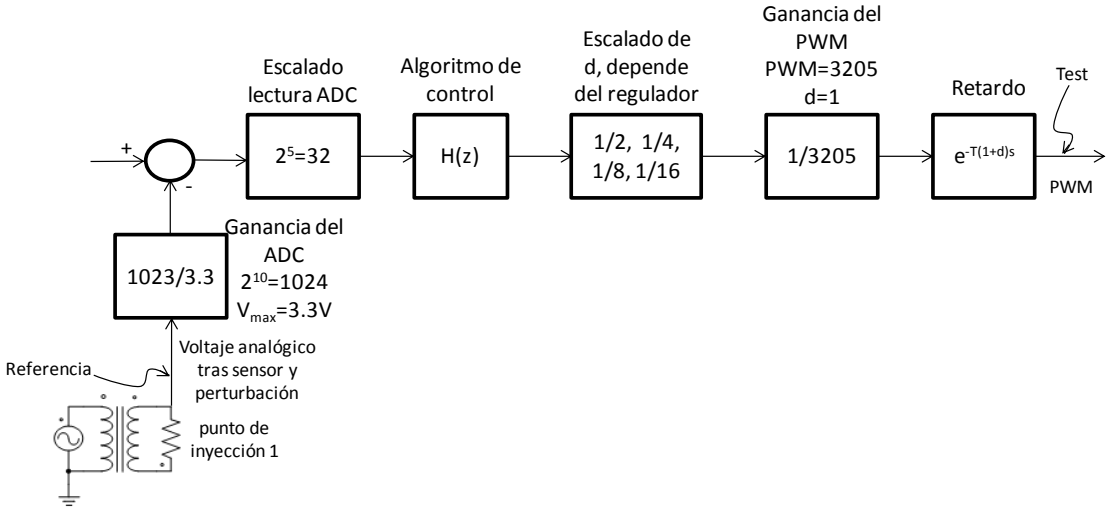


Figura 7-6 Componentes del regulador incluidos en la medida

En la Figura 7-6 se ven todos los bloques que intervienen en el valor final de la medida del regulador entre los puntos Referencia (tensión analógica tras sensor de V_{out} y la

perturbación flotante) y Test (PWM entre 0V y 3,3V) accesibles. A continuación se va a describir uno por uno los bloques incluidos en la medida del regulador:

7.1.2.2.1 Ganancia del ADC

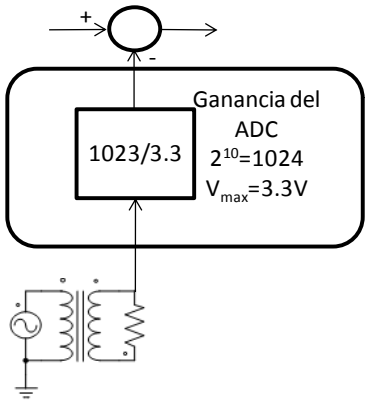


Figura 7-7 Ganancia introducida por el ADC

Se trata de la ganancia que introduce el ADC. En este caso se tiene que si a la entrada del ADC, que está integrado en el dsPIC, está en su valor máximo, 3,3V, se tiene una salida del ADC igual a todos sus bits a "1", que con una resolución de 10 bits.

$$2^{10} = 1024$$

Por lo tanto 0V sería 0 en lectura del ADC y 3,3V serían 1023 en lectura del ADC, por lo tanto este bloque introduce una ganancia de:

$$\mathrm{G_{ADC}} = \frac{1023}{3{,}3}$$

7.1.2.2.2 Escalado de lectura del ADC

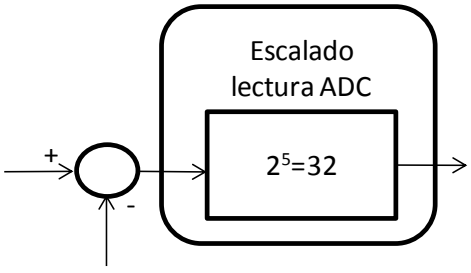


Figura 7-8 Ganancia introducida por el escalado de la lectura del ADC

En este caso el programa del dsPIC introduce una ganancia también positiva ya que escala el valor leído por el ADC en x32. Es decir, a nivel de bits, desplaza el valor contenido en el registro de lectura del ADC en 5 posiciones. Esto se hace porque el dsPIC maneja registros de 16 bits y el valor leído del ADC es de 10. Por lo tanto se escala corriendo los bits en 5 posiciones a la izquierda para que el valor leído, se posicione en la zona más significativa del registro donde está.

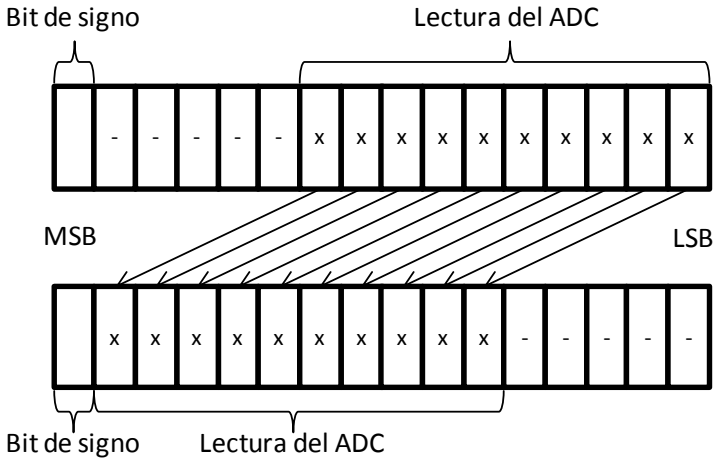


Figura 7-9 Escalado en la lectura del ADC

Este desplazamiento hacia la izquierda de los bits de la lectura del ADC, en valores decimales se convierte en una ganancia positiva de x32.

$$G_{\text{escaladoADC}} = 2^5 = 32$$

7.1.2.2.3 El algoritmo de control

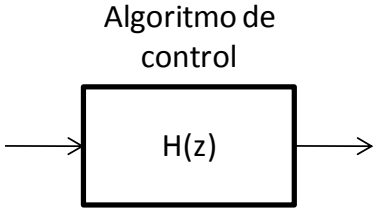


Figura 7-10 Ganancia del algoritmo de control, el regulador en si mismo

Es el único que varía su comportamiento en función de la frecuencia, tanto para la ganancia como para la fase. Proviene del cálculo que realiza el dsPIC mediante multiplicaciones y sumas para construir la ecuación en diferencias del control, tanto con los coeficientes del denominador como con los del numerador de la FDT. Este regulador es el que tiene la misma respuesta en frecuencia que el que se diseño con SmartCtrl al realizar el proceso de diseño descrito en el apartado 5.1.

7.1.2.2.4 Ganancia del modulador o de "d"

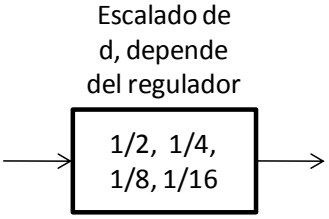


Figura 7-11 Ganancia introducida por el modulador "Ajuste d"

Es una ganancia introducida en el código programado en el dsPIC, que se encarga de ajustar el valor de la señal "d", que proviene del cálculo del algoritmo de control y que va a ser

modulada mediante una señal PWM. Se usa para hacer suficientemente significativa la señal "d" frente a las rampas que generan el PWM. Este nivel de escalado es distinto para cada tipo de control, y su ajuste ha sido mediante prueba y error experimentalmente para conseguir un funcionamiento satisfactorio del control en lazo cerrado (principalmente mediante el comportamiento en régimen permanente para distintos niveles de Vin).

Finalmente en las pruebas experimentales esta ganancia ha sido de:

$$G_{escalado\ d} = {}^{1}/_{2} \text{ ó } {}^{1}/_{4} \text{ ó } {}^{1}/_{8} \text{ ó } {}^{1}/_{16}$$

Valores que han sido ajustados experimentalmente.

7.1.2.2.5 Ganancia del PWM

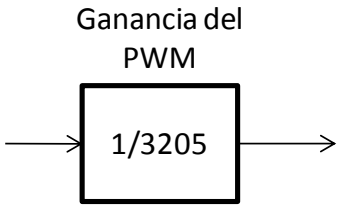


Figura 7-12 Ganancia introducida por el modulador "Pasos PWM"

El último bloque que introduce una ganancia en el regulador medido es el de PWM. En este caso se tiene que la rampa que genera el PWM mediante su comparación con el valor d, tiene 3205 pasos. Este es el número que hace que se consiga los 300kHz de frecuencia de conmutación. Esto supone que si el PWM está a su máximo, siempre con su salida a nivel alto, d debe ser d=3205. En este hipotético caso el PWM estaría siempre a 1, que sería su fondo de escala, con lo que finalmente se introduce con este bloque una ganancia de:

$$\mathrm{G_{PWM}} = {}^{1}/_{3205}$$

7.1.2.2.6 Ganancia final introducida en el bloque Regulador

Finalmente se tiene que de la descomposición del bloque regulador en sus distintas partes que introducen ganancias (el retardo no afecta a la ganancia), se obtiene una ganancia global que se debe añadir a lo medido.

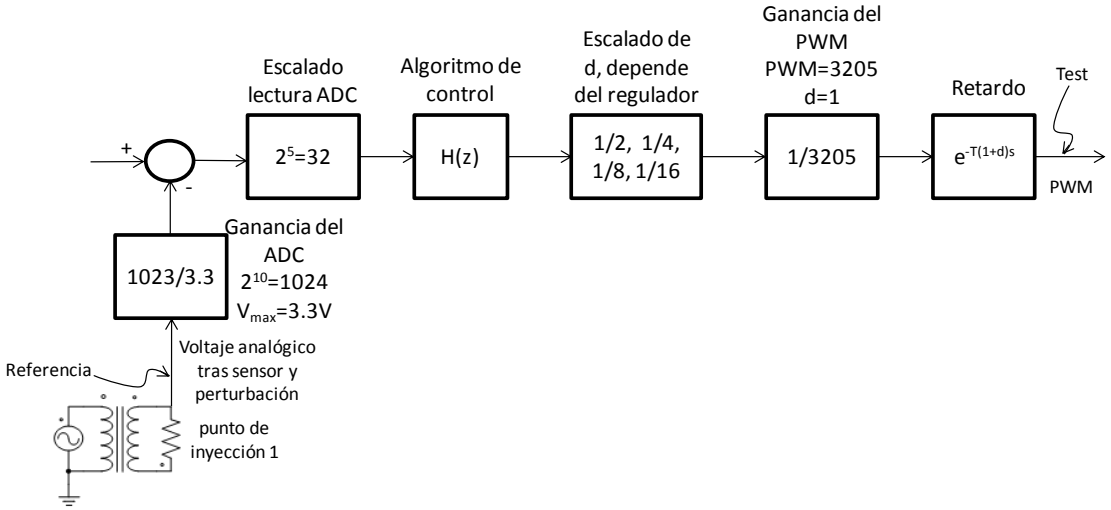


Figura 7-13 Ganancias finales introducidas en la medida del regulador

Los únicos bloques que tienen un comportamiento dependiente de la frecuencia son el retardo, que sólo afecta a la fase y fue introducido en la planta, y el algoritmo de control H(z) que sí afecta en ganancia y fase. Los demás sólo introducen ganancia y no afectan a la fase. Por lo tanto estos bloques de ganancia constante se pueden simplificar en uno que incluya todos y que tendrá la siguiente forma:

$$G_{dsPIC} = G_{ADC} \cdot G_{escaladoADC} \cdot G_{escalado\ d} \cdot G_{PWM} \qquad \textit{Ec. 7-5}$$

$$G_{dsPIC} = {}^{1023}/_{3,3} \cdot 32 \cdot {}^{1}/_{n} \cdot {}^{1}/_{3205} \qquad \textit{Ec. 7-6}$$

Esta ganancia debe añadirse directamente a la del algoritmo H(z) y el retardo y con ello se obtendrá la respuesta en frecuencia real del dsPIC. Por lo tanto finalmente del diagrama de bloques de la Figura 7-13, es posible pasar simplificando los bloques que no dependen de la frecuencia al siguiente diagrama:

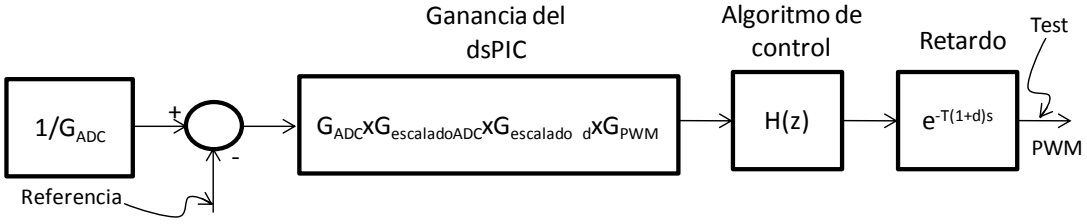


Figura 7-14 Diagrama simplificado de ganancias introducidas

Como experimentalmente se mide la respuesta en frecuencia entre el punto de Referencia y de Test tal y como se ve en la Figura 7-13, se está incluyendo en la medida la ganancia del dsPIC.

En el proceso de diseño descrito en el apartado 5, se acaba obteniendo unos coeficientes de una FDT a introducir en el dsPIC, que son los que forman el algoritmo de control contenidos en el bloque H(z). Por otro lado, a la hora de hacer las medidas, la ganancia Gdspic, debe ser restada a lo medido para obtener una respuesta frecuencial como la que se proporciona SmartCtrl para el bloque regulador. Nótese que esta ganancia Gdspic se introduce en los cálculos de SmartCtrl como parte de la ganancia del sensor, como ya se comento en el

apartado 5.1.2. Además el sensor incluye la constante correspondiente al divisor de tensión resistivo.

Por otro lado, al igual que con la planta, se debe hacer el ajuste de la medida con motivo de sensar en una señal PWM de amplitud 3,3V. A diferencia del caso de la planta, cuya sonda Referencia mide en el PWM (entrada de la planta), ahora es la sonda Test la que queda al PWM (salida del regulador) con lo que:

La entrada (Referencia) mide en la señal de realimentación:

$$X = V_{loop} \qquad \textit{Ec. 7-7}$$

Y a la salida (Test) a la señal PWM:

$$Y = 3{,}3 \cdot d \qquad \textit{Ec. 7-8}$$

Con lo que finalmente la FDT de la salida (ciclo de trabajo) frente a la entrada (Valor de realimentación), queda:

$$G_{dl} = \frac{Y}{X} = \frac{3{,}3 \cdot d}{V_{loop}} = \frac{d}{V_{loop}} \cdot \frac{3{,}3}{1} \qquad \textit{Ec. 7-9}$$

$$20 \cdot log\left(\frac{3{,}3}{1}\right) = +10{,}37dB \qquad \textit{Ec. 7-10}$$

Con lo que la medida está amplificada en 10,37dB por efecto de la modulación que deberá ser restada.

Por consiguiente para obtener las medidas experimentales del regulador, se mide en los puntos accesibles Referencia y Test, se restan 10,37 dB por efecto de la modulación y se resta la ganancia Gdspic para poder comparar el regulador obtenido de SmartCtrl, con lo realmente implementado en el dsPIC, que es lo que se obtiene de la medida. (Sin los 10,37 que no son del dsPIC, sino por el sistema de medida)

7.1.2.3 El lazo

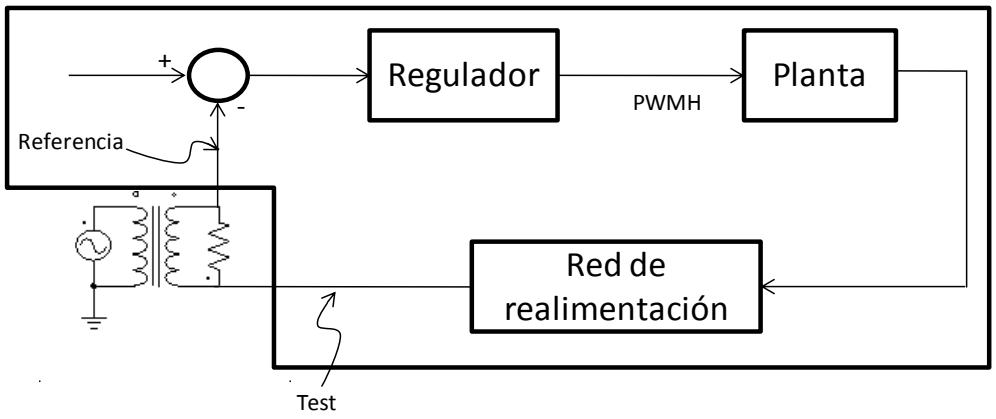


Figura 7-15 Esquema de medida del lazo

Por último, se describe en este apartado la medida de la ganancia de lazo abierto ("*open loop gain*"), que es la ganancia de todos los elementos del lazo de control funcionando en lazo cerrado. En este caso se mide la respuesta en frecuencia en bornas de la resistencia de inyección. Por lo tanto en este caso, no hay que introducir ninguna ganancia ya que la relación entre referencia y test puede hacerse directamente con el analizador de ganancia y fase. En

este aspecto la medida del lazo resulta sencilla, pero es especialmente importante porque al diseñar con SmartCtrl, la frecuencia de cruce (Fc) y el margen de fase (MF) que fija el diseñador, debe verse para este caso directamente. De este modo, en el resultado de esta medida la Fc y el MF deben ser los mismos que los fijados por el diseñador en el apartado 5.1.5, donde se seleccionan en el mapa de soluciones de SmartCtrl la Fc y el MF.

7.1.2.4 El sensor

El sensor no ha sido medido en todos los casos de las medidas experimentales del apartado 8. Sin embargo sí se ha medido para poder contrastar los resultados de la medida directa del lazo, con la suma de las medidas de planta, regulador y sensor. Este comportamiento en frecuencia del sensor no fue introducido en SmartCtrl para los casos de PI y tipo2, y por este motivo los resultados medidos y los que se obtiene de SmartCtrl difieren a alta frecuencia.

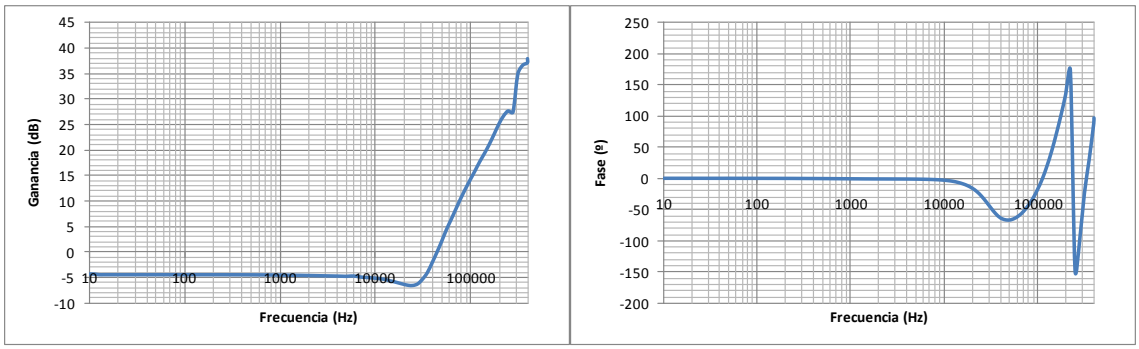


Figura 7-16 Ganancia y fase del sensor

A este fenómeno del sensor a alta frecuencia, se harán constantes referencias en los apartados del punto 8.3, ya que afecta en gran medida a los resultados experimentales.

Para el caso de los lazos de tipo3, este fenómeno de alta frecuencia sí se ha tenido en cuenta en los diseños en SmartCtrl, y previsiblemente los resultados experimentales deberán parecerse más a los teóricos. Sin embargo esto se comprobará más detalladamente en el apartado 8.3.3.

7.1.3 Herramientas de medida

En este apartado se van a describir las herramientas usadas para realizar las medidas experimentales y como han sido usadas para este fin. Principalmente para medidas sólo se han usado los osciloscopios del laboratorio y los analizadores de ganancia y fase. El analizador Venable 3235 se ha usado mayoritariamente para las medidas finales y el HP4194A para las primeras pruebas. En ambos casos con su transformador para inyección de la perturbación.

7.1.3.1 El analizador

Se trata del encargado de generar la perturbación, y medir los resultados entre la entrada y la salida del bloque a medir. Es el que nos permite obtener de forma experimental la respuesta en frecuencia de los distintos bloques a estudiar. En este proyecto se ha usado tanto el Venable 3235 como el HP 4194A, pero las medidas finales contenidas en el apartado de resultados 8, se han realizado con el Venable 3235, excepto las comentadas en el apartado 8.3.3.3 (hechas con el HP 4194A). Sin embargo otras muchas medidas y pruebas se han hecho

con el HP 4194A, ya que el Venable 3235 no estaba en el laboratorio cuando se empezaron a necesitar este tipo de medidas.

Figura 7-17 Analizador HP 4194A

Figura 7-18 Analizador Venable 3235

7.1.3.2 El "*bode box*"

El "*bode box*" es un transformador 10:1 que se usa para inyectar la perturbación que genera el analizador a través de su salida "*oscillator*" en el lazo, a través de una resistencia de 100Ω, en el jumper correspondiente de la placa buck/boost. Mediante este método se consigue inyectar una perturbación flotante en el lazo. En caso del "*bode box*" usado en este proyecto, el mismo fabricante recomienda usarlo para frecuencias entre 100Hz y 10kHz. Esto es así porque las inductancias y capacidades internas del "*bode box*", están dimensionadas para tener a esas frecuencias baja corriente magnetizante y desacoplo entre primario y secundario, quedando que fuera de ese rango de frecuencias el transformador de inyección puede afectar a las medidas.

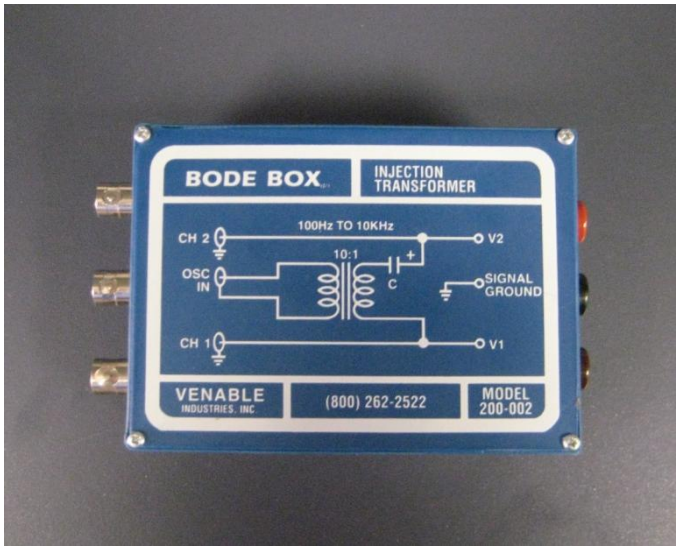


Figura 7-19 "Bode Box"

7.1.3.3 Osciloscopio

Para las medidas de respuesta temporal se han usado osciloscopios del laboratorio. Estas medidas han sido usadas en primer momento para ver como el dsPIC regulaba el PWMxH, al forzar variaciones en la tensión de entrada del convertidor mediante un montaje como el que se ve en la Figura 4-14. También se ha usado osciloscopio para poder medir cuanto tiempo tarda el dsPIC en modificar el PWM de un valor anterior a uno nuevo, lo que implica realizar todo el cálculo de control y además activar el cambio sobre la señal PWM.

Por otro lado y como apoyo de las medidas en frecuencia, se han usado osciloscopios midiendo tensión a la salida del convertidor para ver como llegaba la perturbación a la carga. De este modo se puede ver la perturbación a las distintas frecuencias del barrido y su atenuación. Sin embargo esta medida era simplemente orientativa, ya que siempre se media en la salida del convertidor, y sólo con una medida de la perturbación, con lo que no es posible obtener ganancia y fase. Estas medidas mostraban si realmente en la carga la perturbación era senoidal o si presentaba armónicos, los cuales aparecían especialmente usando amplitudes de la perturbación grandes. Estas grandes amplitudes pueden provocar no linealidades y frecuencias distintas de la principal de la perturbación inyectada. Como ejemplo ilustrativo de este tipo de medidas de osciloscopio, véase más adelante la Figura 7-20, donde en el osciloscopio se puede ver la frecuencia y amplitud de la perturbación introducida en el lazo, en la carga del convertidor.

7.1.3.4 PC

Se trata de un PC portátil especialmente configurado para realizar las medidas con el analizador Venable que es el que ha sido usado mayoritariamente. Este PC dispone de un software específico proporcionado por el fabricante del analizador Venable, que permite fácilmente realizar las medidas de ganancia y fase que se han tomado en este proyecto, donde su mayor virtud consiste en la facilidad y rapidez en extraer los datos de la medición. De nuevo sobre la Figura 7-20, se puede ver el ordenador portátil y la grafica que está tomando de la planta a través del analizador.

7.1.4 Montaje experimental

Finalmente en este apartado se va a describir el montaje final sobre el que se han hecho todas las medidas experimentales. Como se ve en la Figura 7-20, todas las herramientas descritas en el apartado 7.1.3, son usadas en este montaje final para medidas de respuesta en frecuencia.

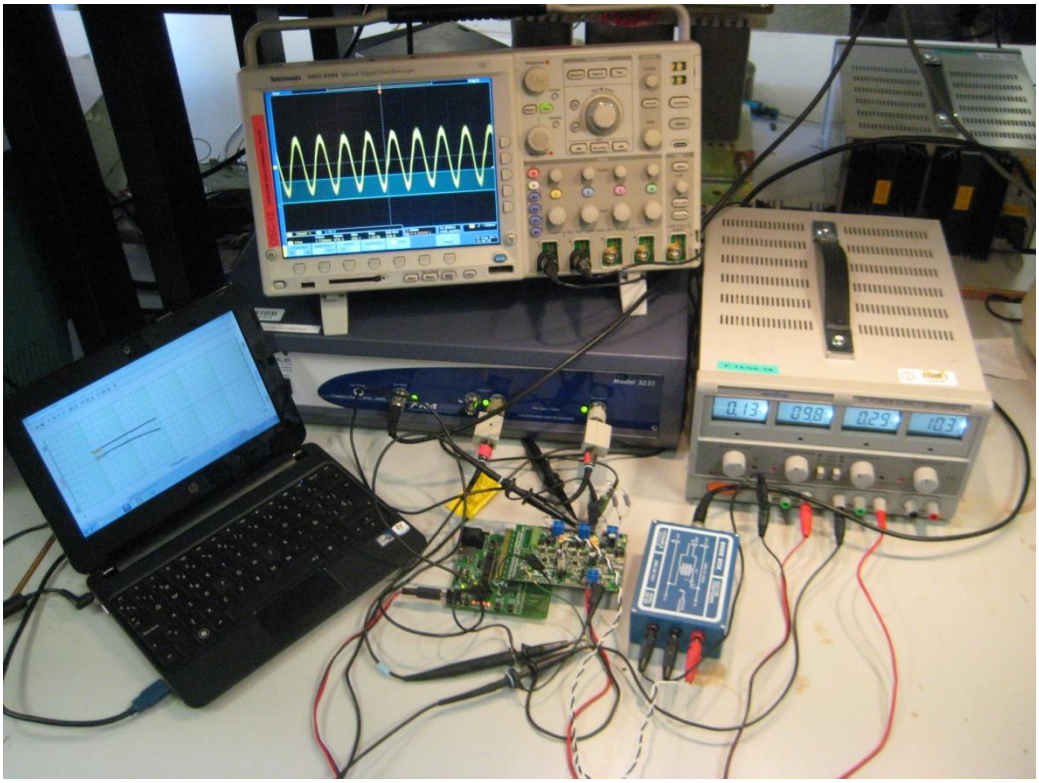


Figura 7-20 Montaje para medida de respuesta frecuencial

En esta foto se está realizando un barrido en frecuencia para la medida de la planta, donde se ven las placas *Buck*/*Boost* y la placa de control, alimentadas por una fuente de continua, el *"bode box"*, que permite introducir la perturbación flotante en el lazo, el analizador, el PC y el osciloscopio.

El analizador es quien genera la perturbación, y el que hace la medida mediante sus sondas de referencia y test, que en este caso están entre el pin PWM2xH (pulso de control del transistor principal del reductor2) y la carga del reductor2 (tensión de salida del mismo). El analizador es quien envía esta información de la relación entre entrada y salida para ganancia y fase al PC, que mediante el software de Venable, muestra los resultados en pantalla como se ve en la foto. Por último sobre el osciloscopio se ve en el canal 1 (amarillo) como varía la tensión AC de salida del reductor por efecto de la perturbación introducida en el lazo. En el canal 2 (azul) se ve la señal PWM2xH, aunque no se puede apreciar mucho porque la frecuencia de conmutación es mucho mayor que la frecuencia de la perturbación que se ve en amarillo en este instante de la foto.

En la Figura 7-21, se ve el detalle de las placas *Buck*/*Boost*, la placa de control, el "*bode box*" y dos sondas del analizador Venable 3235, dispuestas para la medida del regulador y su correspondencia con el esquema simplificado del lazo que se ha usado en la redacción de este proyecto en la mayoría de los casos.

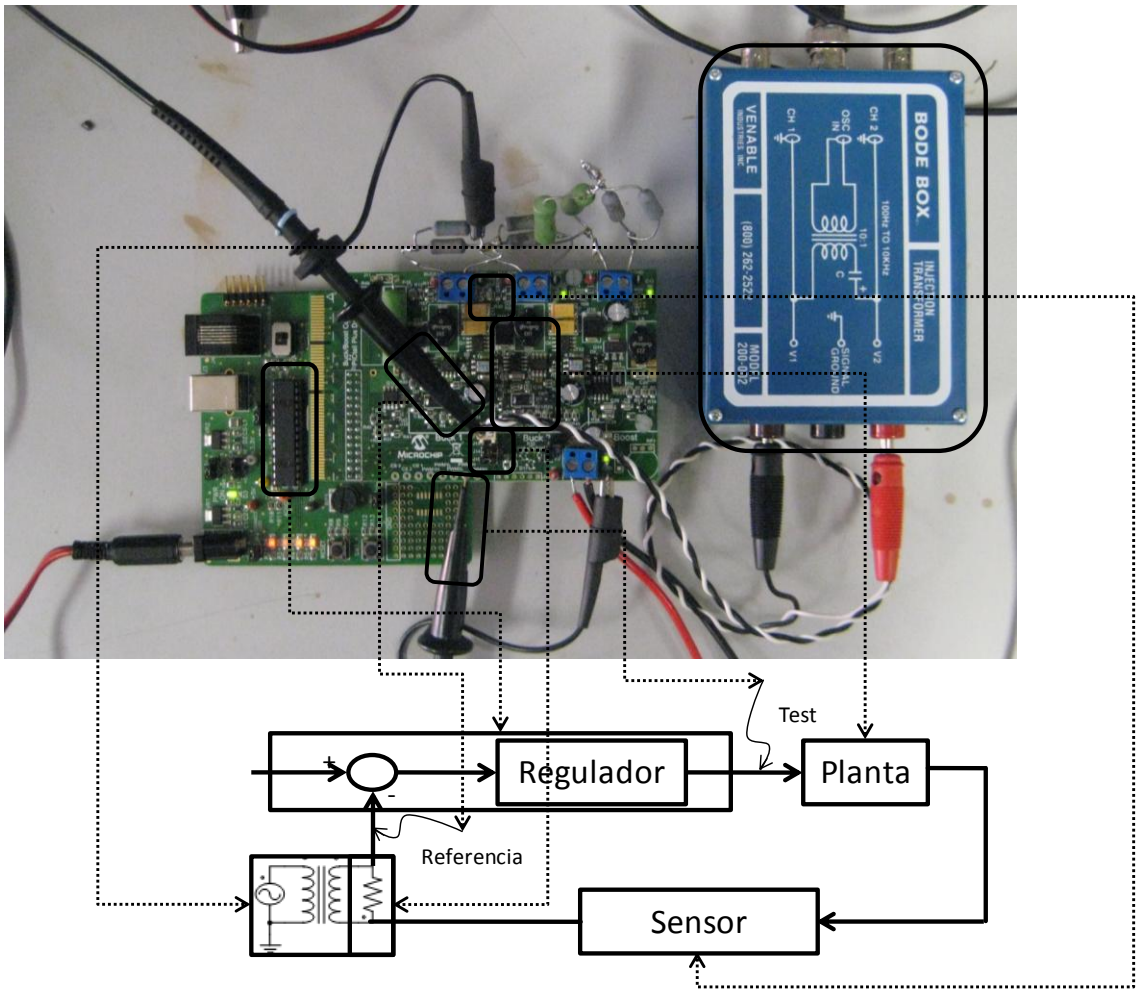


Figura 7-21 Detalle de planta, regulador y transformador de inyección para medida de respuesta frecuencial y su figura equivalente

En esta foto-esquema, se ve el montaje real para el caso de medida del regulador y su equivalencia en el esquemático. En él se pueden ver las sondas de Test y Referencia reales y su posición de medida en el esquema. Se muestra también el "*bode box*", que es el transformador que inyecta la perturbación flotante, y conectado a él por los cables blanco y negro, la resistencia de inyección. También se muestra sobre la foto los componentes principales del reductor2 y junto a estos, la red de realimentación o sensor, que es un divisor de tensión resistivo ubicado junto a la salida del reductor2. Finalmente también se muestra el dsPIC ubicado en la placa de control y que sobre el esquemático engloba al regulador y al restador de referencia y valor de realimentación.

En la siguiente foto-figura se muestra el mismo esquema pero ahora más simplificado, sin las sondas para la medida, ni el *"bode box"*, pero si con la resistencia de inyección de la perturbación. En este esquema se ve de una manera sencilla la equivalencia entre los componentes del lazo mostrados en la foto y los bloques de los esquemáticos mostrados en esta memoria.

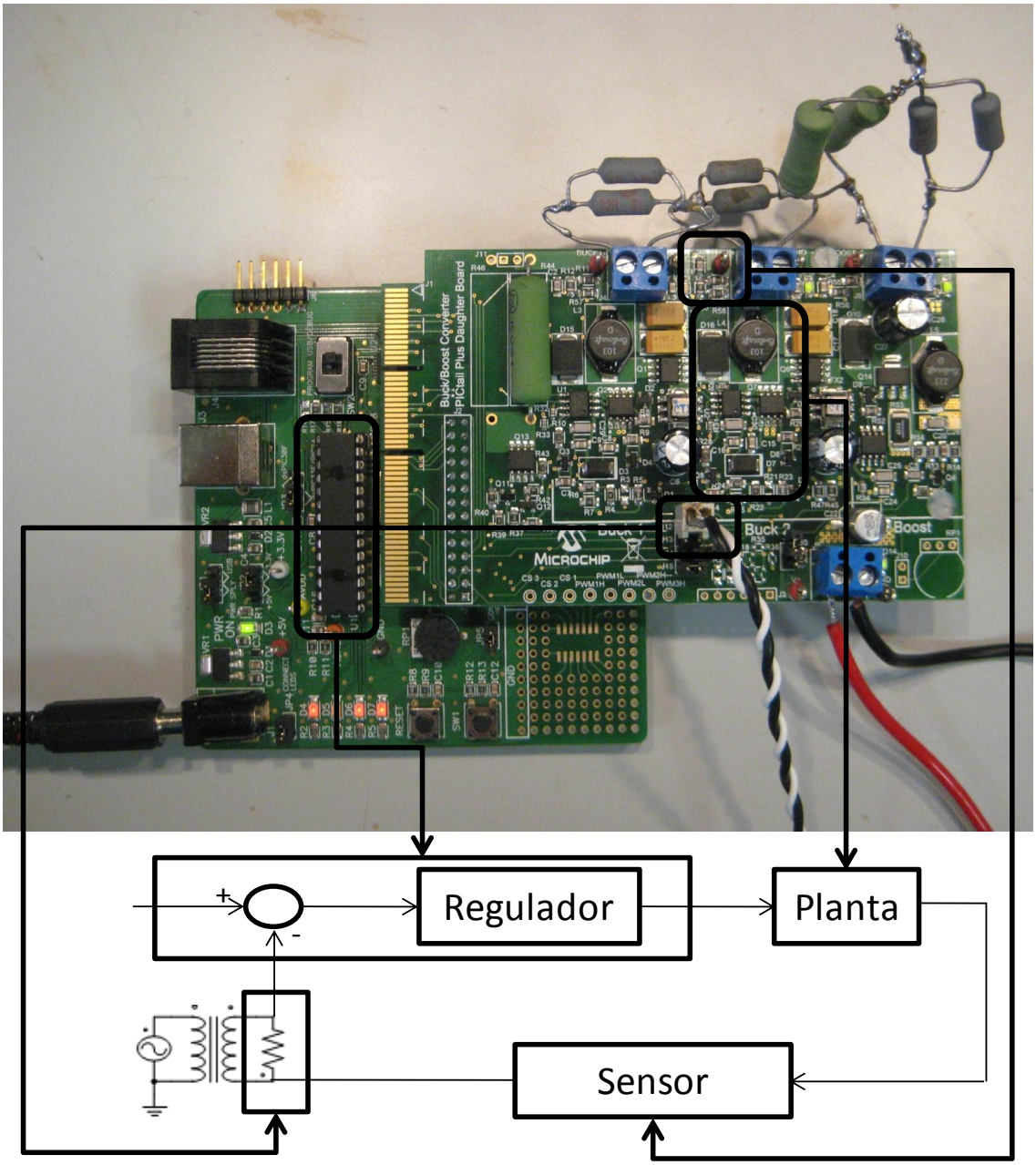


Figura 7-22 Detalle de las placas de convertidores y de control

8 Resultados experimentales

Se han hecho multitud de medidas experimentales sobre el montaje programado y funcionando en lazo cerrado, aunque se ha dado especial importancia a la respuesta frecuencial del lazo y sus componentes. Por lo tanto en este capítulo se va a detallar el proceso y las medidas obtenidas para validar los resultados experimentales, y con ello el proceso de diseño de reguladores digitales con SmartCtrl, centrándose en la respuesta en frecuencia del sistema completo. Cabe recordar que el objetivo principal de este proyecto es conseguir que siguiendo los pasos descritos en el apartado 5, y realizando las medidas experimentales tal y como se indica en el apartado 7, sea posible diseñar e implementar un control tal y como el diseñador quiera.

8.1 Características generales de las medidas experimentales

Para las medidas de respuesta en frecuencia se han hecho barridos desde 10Hz hasta 400kHz con amplitudes en la perturbación que dependían del caso. Con esto se obtienen medidas experimentales desde muy baja frecuencia, lo cual ha dado muchos problemas en la medida, ya que a frecuencias bajas la toma de datos es muy lenta y además en algunos casos problemática. Respecto a la amplitud de la perturbación introducida, normalmente se ha seleccionado aquella que era mayor, sin provocar no linealidades en el lazo, es decir manteniendo las condiciones de pequeña señal que permiten linealizar la dinámica de un sistema en torno a un punto del que no se va a alejar mucho, donde una perturbación muy amplia puede, sacar al sistema de su zona lineal. Sin embargo pese a este problema, las medidas experimentales salen mejor con amplitudes grandes, ya que las medidas en los canales Test y Referencia son más grandes, lo que ayuda al analizador a obtener la relación de amplitud y fase a la frecuencia correspondiente.

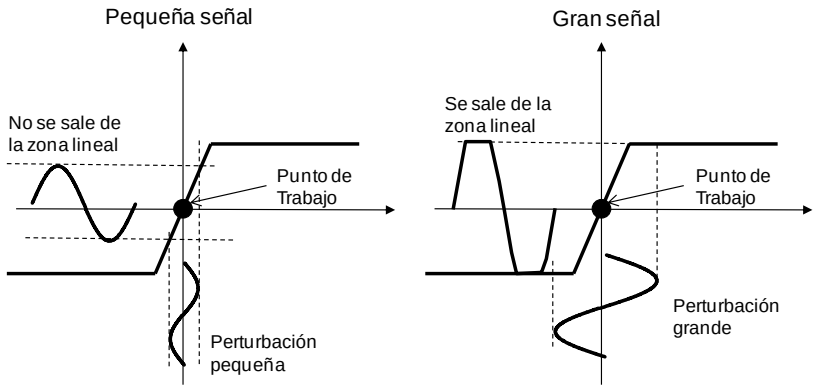


Figura 8-1 Perturbación como pequeña o gran señal

Las medias de respuesta en frecuencia experimentales obtenidas para este proyecto, han sido tomadas con el Venable 3235 (excepto Tipo3_3, que se han medio con el HP 4194A) y el "*bode-box*" 100Hz-10kHz, descritos en el Capítulo 6. Esto supone que las medidas son óptimas para ese rango de frecuencias por lo que al "*bode-box"* se refiere. Aunque el barrido se sale del rango del *"bode-box"*, se ha comprobado que este no es el principal problema.

El principal problema en la medida, está para frecuencias bajas en reguladores y lazos, no en la planta. Esto es así porque tanto el regulador como el lazo parten desde inicio con un

polo en el origen, lo que en ganancia es bajar con una pendiente de -20dB/dec desde 0 Hz. Esto supone que a frecuencia 0 la ganancia es ∞, y en general que para frecuencias bajas la ganancia es muy alta. Esto es un problema para el analizador, ya que al haber una ganancia muy alta, el nivel de la señal que se mide varía mucho entre los puntos de medida Test y Referencia, donde esa diferencia en amplitud es enorme y tiene problemas para sacar la ganancia entre una señal muy pequeña y otra muy grande. Por este motivo, se estima que las medidas con mucha ganancia, es decir en frecuencias bajas para regulador y lazo por tener polo en el origen, pueden dar resultados inesperados en algunos casos.

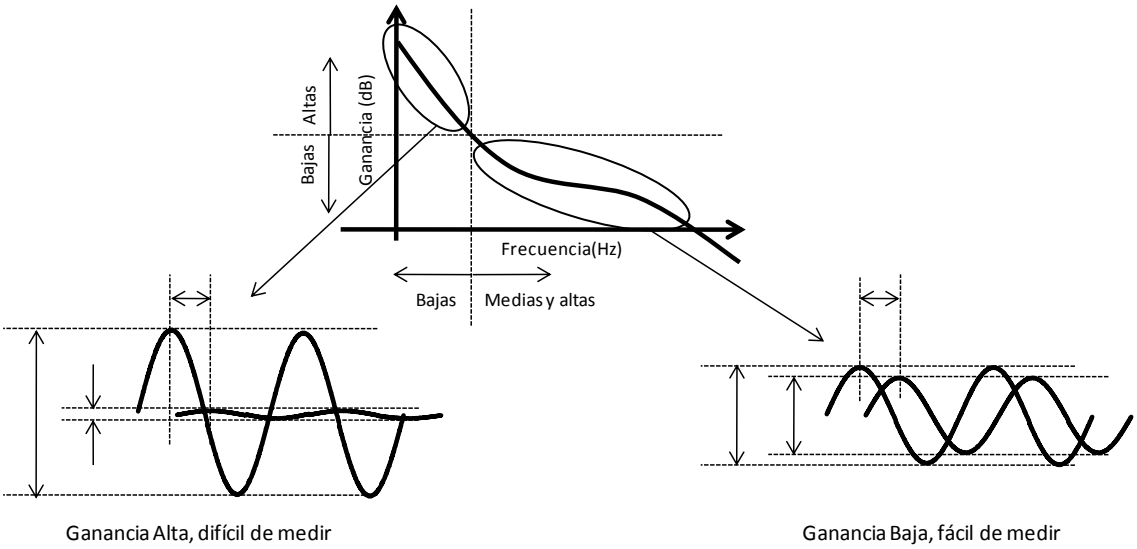


Figura 8-2 Medida con integrador en altas o bajas frecuencias

Este problema no se da en la planta, ya que esta para bajas frecuencias, antes de la resonancia del filtro de salida del convertidor, tiene un comportamiento en frecuencia plano, sin tener ganancias muy grandes a ninguna frecuencia. Por lo tanto, para frecuencias bajas (experimentalmente de 10 a 100Hz) los resultados de las medidas para regulador y lazo pueden dar resultados inesperados, pero para frecuencias mayores, los resultados deberían ser muy similares a los valores teóricos de SmartCtrl y MatLab.

8.2 Medidas teóricas con las que se comparan los resultados experimentales

Para la implementación y validación del proceso de diseño de reguladores digitales, las medidas obtenidas experimentalmente se comparan con unos valores teóricos obtenidos desde SmartCtrl y MatLab para el caso del regulador. Para cada bloque, independientemente del tipo de control, se han hechos las siguientes comparaciones entre lo teórico y lo experimental.

8.2.1 Planta

La comparación entre la planta teórica y la experimental se hace simplemente entre la planta medida y la considerada como teórica que tiene el SmartCtrl. Sin embargo hay que recordar que la planta de SmartCtrl fue importada de otra medida experimental de la planta,

por lo que finalmente se tiene la comparación entre dos medidas experimentales, una hecha con un control genérico y la otra con el regulador concreto a estudio. Por lo tanto finalmente queda la comparación de dos medidas experimentales distintas tomadas con distintos controles, donde es de suponer que ambas medidas serán muy similares.

Por otro lado hay que recordar que en la medida experimental, no hay ningún retardo digital y en la planta que se importó, si se añadió el retardo del lazo digital, aunque realmente no pertenece a la planta, sino al regulador. Por lo tanto la planta "teórica", la de SmartCtrl, y la experimental, en fase deben diferir por el efecto del retardo digital, que será más significativo para frecuencias altas.

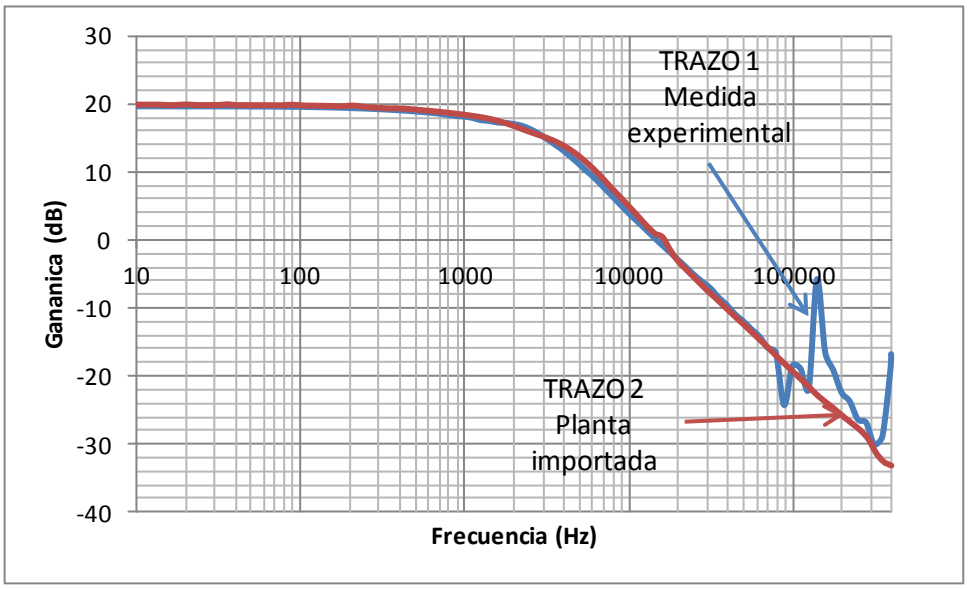


Figura 8-3 Ejemplo de ganancia de la planta

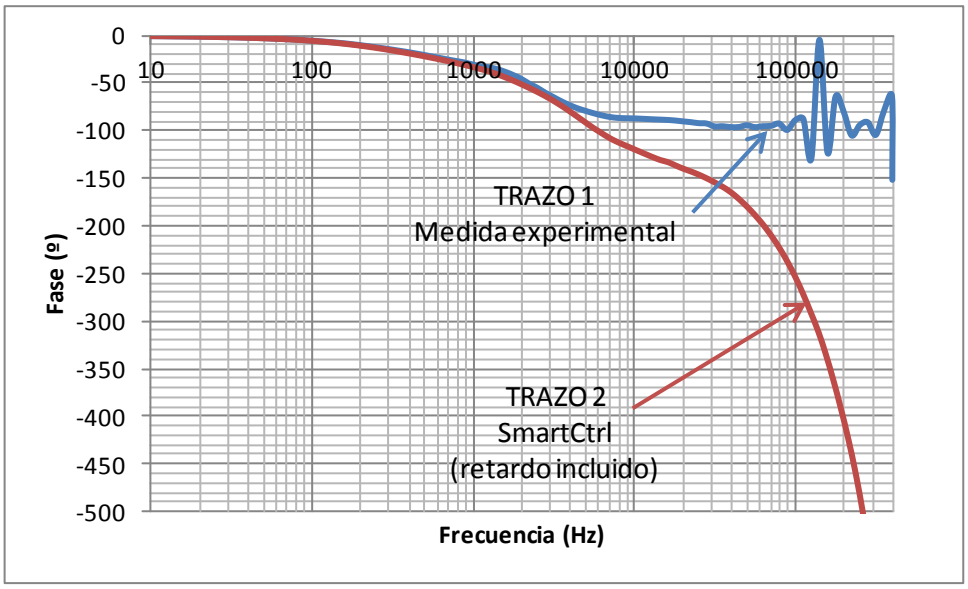


Figura 8-4 Ejemplo de fase de la planta

8.2.2 Regulador

Para la comparación teórica y experimental del regulador, se ha usado SmartCtrl, MatLab y la medida. Mediante SmartCtrl se obtiene un regulador continuo que cumple las especificaciones de Fc y MF fijadas por el usuario para el diseño del lazo, y que es exportado para poderse comparar con las otras dos medidas. Por otro lado los coeficientes de la FDT continua que nos da SmartCtrl, se introducen en un script de MatLab que discretiza el regulador, genera con ellos una FDT discreta y obtiene la respuesta en frecuencia de ese regulador digital. De este modo se tiene el mismo regulador de SmartCtrl pero discretizado, donde los coeficientes discretos tienen la resolución de MatLab, es decir, no están limitados por el formato Q15 que maneja el dsPIC (la resolución de MatLab es mucho mayor que la del dsPIC con Q15 y se considerará ilimitada).

Finalmente la medida experimental es comparada con los resultados procedentes de SmartCtrl (regulador continuo, resolución ilimitada) y los procedentes del MatLab (regulador discreto, resolución ilimitada). La medida experimental es la de un regulador digital con una resolución limitada por manejar números en formato Q15 (regulador discreto, resolución limitada Q15, 15 bits).

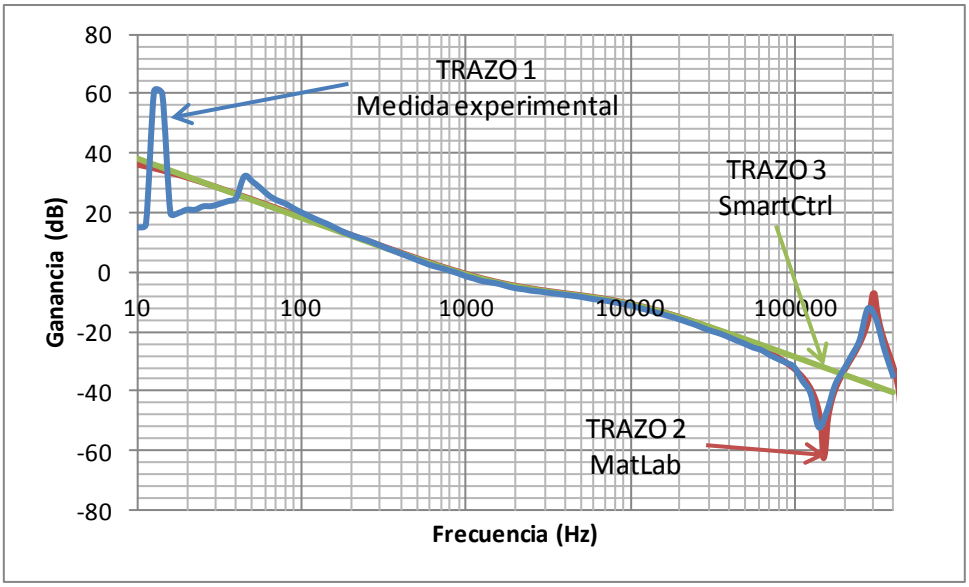


Figura 8-5 Ejemplo de ganancia del regulador

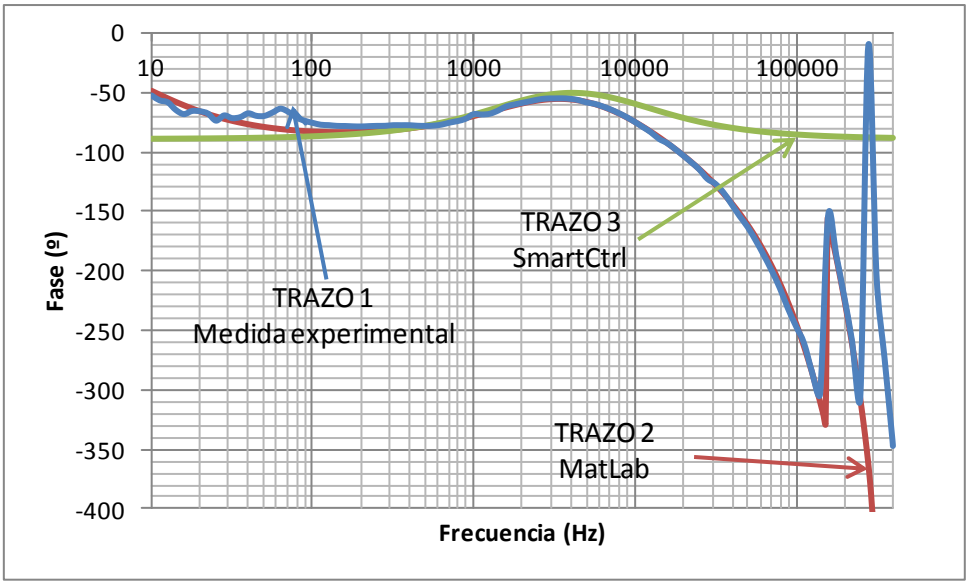


Figura 8-6 Ejemplo de fase del regulador

8.2.3 Lazo

Para la comparación entre lo teórico y lo experimental se ha extraído de SmartCtrl la ganancia y fase del lazo como medida teórica y se ha comparado con la medida experimental del lazo por un lado y con la suma de medidas experimentales de los distintos bloques integrantes del mismo (planta, regulador y sensor) obtenidos experimentalmente por otro lado. Por lo tanto se tienen tres curvas distintas:

- Una teórica de SmartCtrl (llamada "SmartCtrl" en las figuras),
- Una experimental directamente sobre el lazo ("medida experimental simple") ,
- Y otra experimental como suma de los componentes del lazo ("medida experimental suma").

En estas medidas del lazo, es donde se debe ver como las especificaciones de Fc y MF fijadas en el proceso de diseño con SmartCtrl descrito en el apartado 5.1, se cumplen razonablemente en las medidas experimentales.

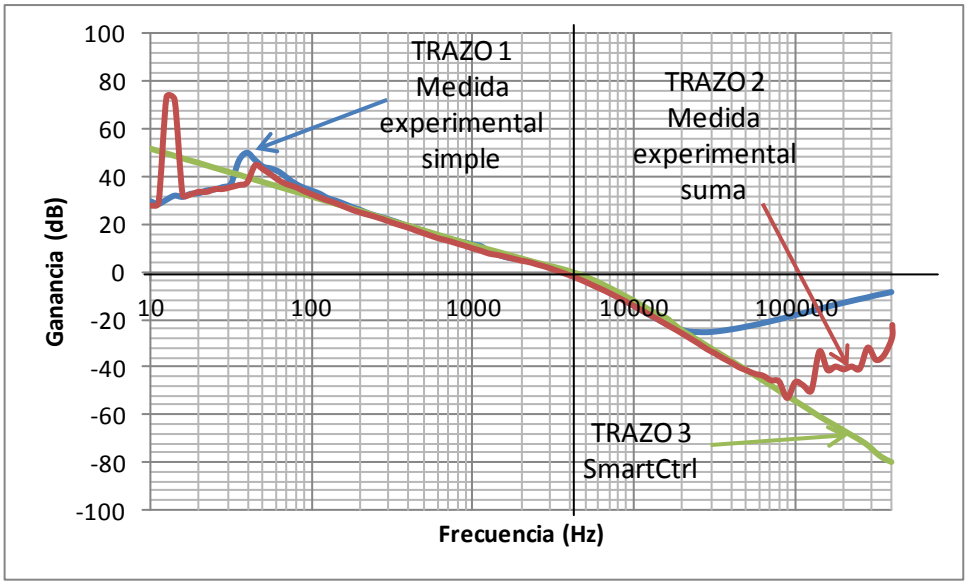


Figura 8-7 Ejemplo de ganancia del lazo abierto

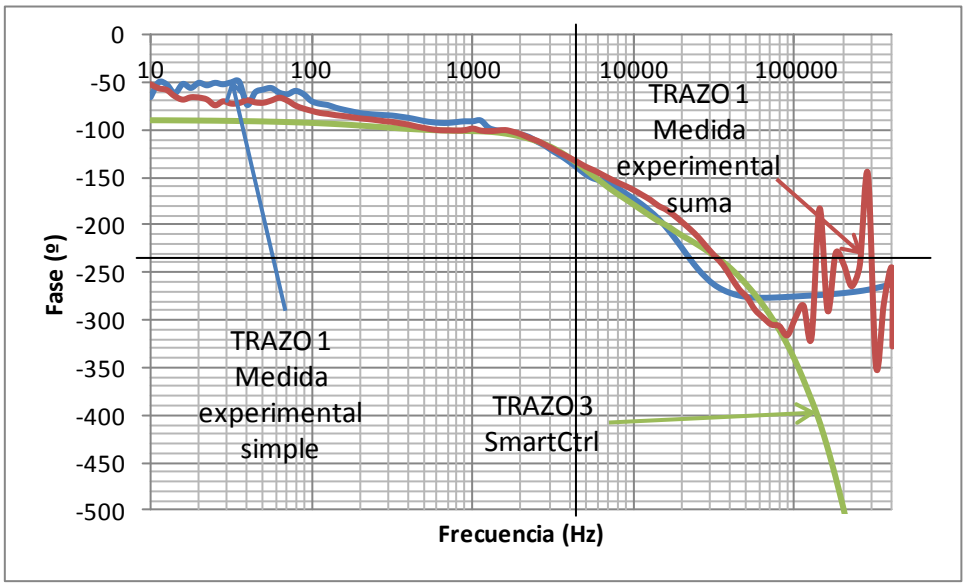


Figura 8-8 Ejemplo de fase del lazo abierto

8.3 Metodología de obtención de medidas experimentales

Como se describe en el apartado 5, mediante SmartCtrl es posible diseñar lazos de control mediante la selección de un tipo de regulador y sus características de frecuencia de cruce (Fc) y margen de fase (MF). Por lo tanto las medidas experimentales para validar el proceso de diseño de este proyecto consisten en el diseño de 9 reguladores, de los cuales 3 son de tipo PI, 3 son tipo2 y los 3 últimos tipo3. Dentro de cada tipo de regulador se ha tratado de tener unas Fc y MF distintos entre sí, para poder justificar el funcionamiento del sistema para una zona lo más amplia posible del mapa de soluciones que ofrece el SmartCtrl. Por ello, las medidas experimentales se desglosaran en los tres tipos distintos de control, cada uno con

sus particularidades y explicaciones, y después se mostrarán los resultados específicos de cada uno de los tres controles del mismo tipo.

8.3.1 Reguladores tipo PI

En este grupo están los tres reguladores de tipo PI, cuyas FDT tienen la forma:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_1 s+1}{a_1 s} \qquad \textit{Ec. 8-1}$$

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0+b_1 z^{-1}}{1+a_1 z^{-1}} \qquad \textit{Ec. 8-2}$$

Los Márgenes de Fase y Frecuencia de corte seleccionados para este tipo de control, son los que se ven en la figura siguiente:

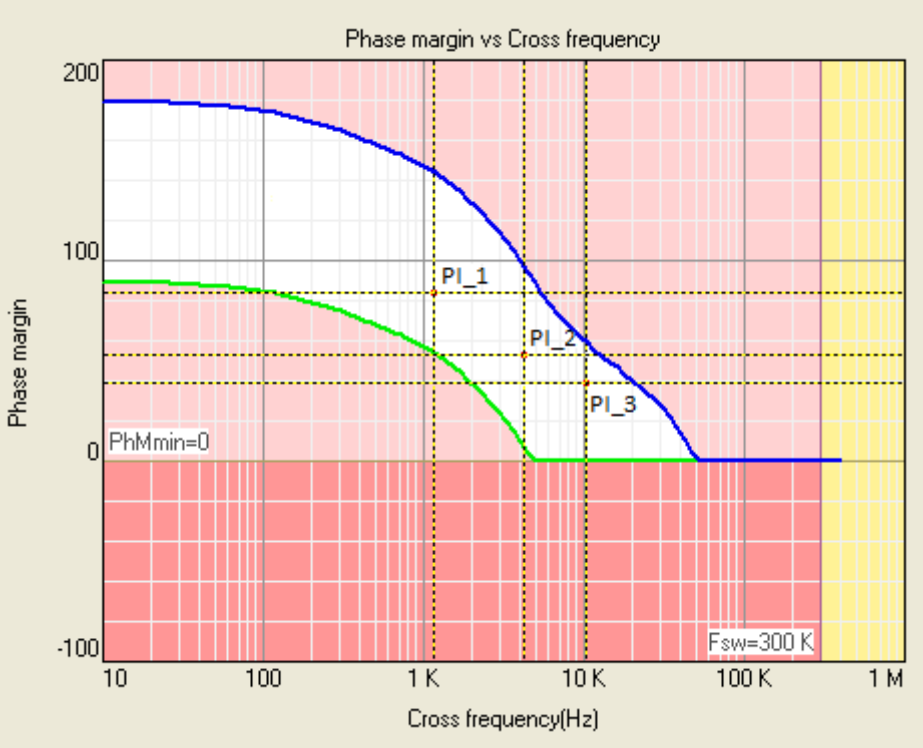


Figura 8-9 Mapa de soluciones para controles PI

En Figura 8-9 se pueden ver juntos los tres puntos del mapa de soluciones que se han implementado en el montaje real, para obtener las medidas experimentales que se detallaran a continuación. Como resumen se tienen los siguientes reguladores:

	PI 1	PI 2	PI 3
Fc (kHz)	1,15478	4,21697	10,227
MF (º)	84	53	39,276

Tabla 8-1 Frecuencia de cruce (fc) y margen de fase (MF) para lazos de tipo PI

8.3.1.1 PI_1

En este primer caso se tiene un control que fija la Fc antes de la frecuencia de resonancia del filtro de salida del convertidor (2900Hz aproximadamente) y un Margen de Fase alto:

Frecuencia de cruce (Fc):	1,15478kHz
Margen de Fase (MF):	84º

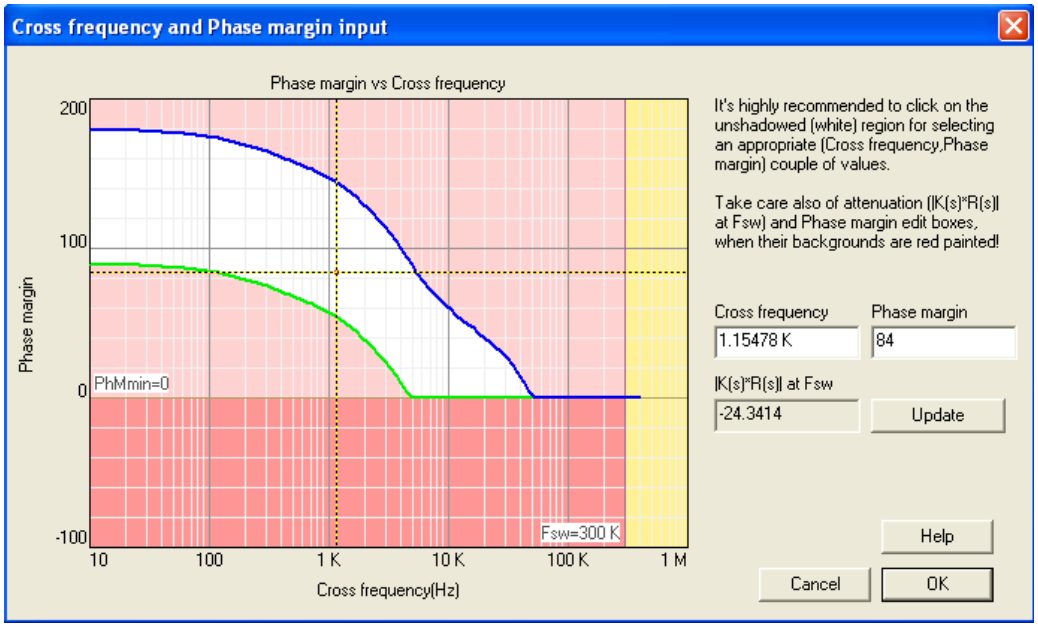


Figura 8-10 Mapa de soluciones PI_1

Para estas condiciones el sensor implementado en SmartCtrl tal y como se indica en el apartado 7.1.2.2.6, ha sido el siguiente:

$$G_{dsPIC} \cdot G_{sensor} = \frac{1}{8} \times 32 \times \frac{1023}{3{,}3} \times \frac{1}{3205} \times \frac{5}{8{,}3} = 0{,}23307 \qquad \textit{Ec. 8-3}$$

8.3.1.1.1 Planta PI_1

Los resultados de la planta han sido los siguientes:

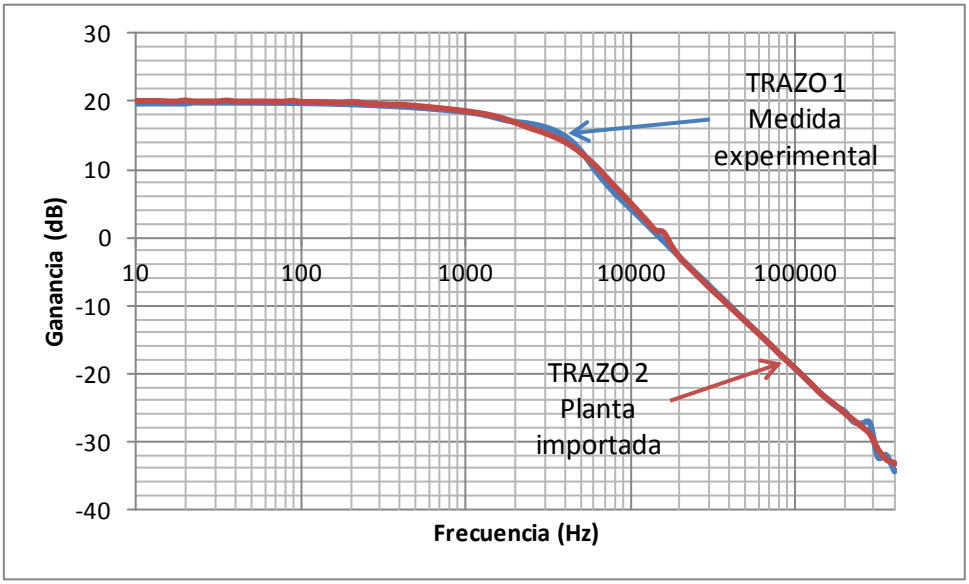


Figura 8-11 Resultados de la ganancia de la planta PI_1

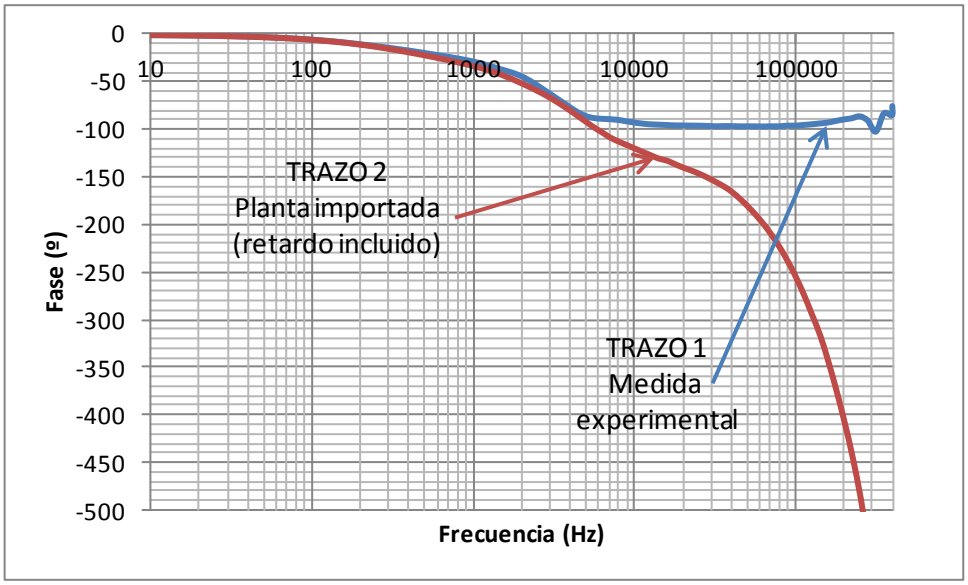


Figura 8-12 Resultados de la fase de la planta PI_1

Con carácter general los resultados obtenidos son coincidentes en ganancia. Para la fase, el efecto del retardo introducido en la planta importada hace que esta caiga con la frecuencia, quedando que ha frecuencias bajas y medias, la planta medida y la importada resultan razonablemente coincidentes, pero para altas no por efecto del retardo.

8.3.1.1.2 Regulador PI_1

Los resultados del regulador implementado en este primer caso de PI, son los siguientes:

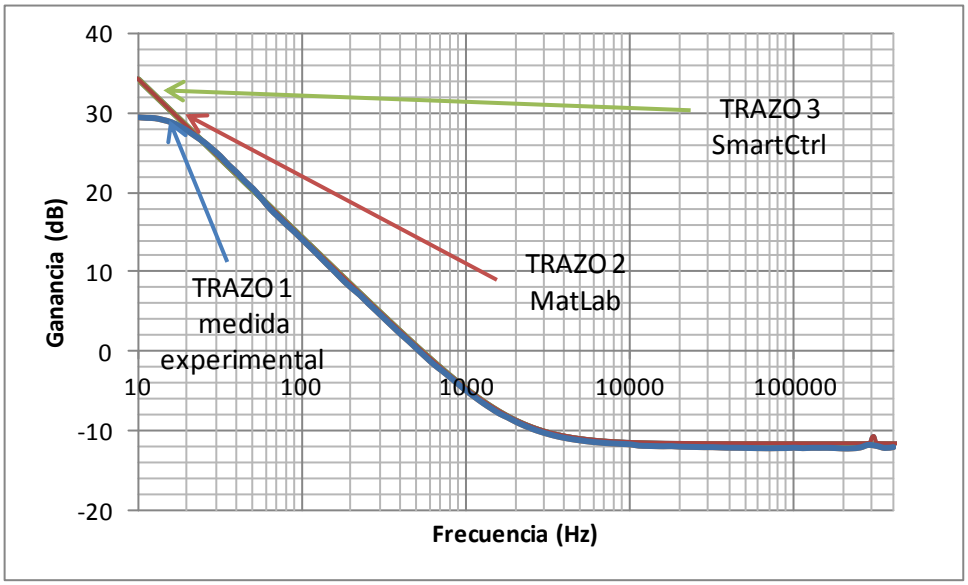


Figura 8-13 Resultados de ganancia del regulador PI_1

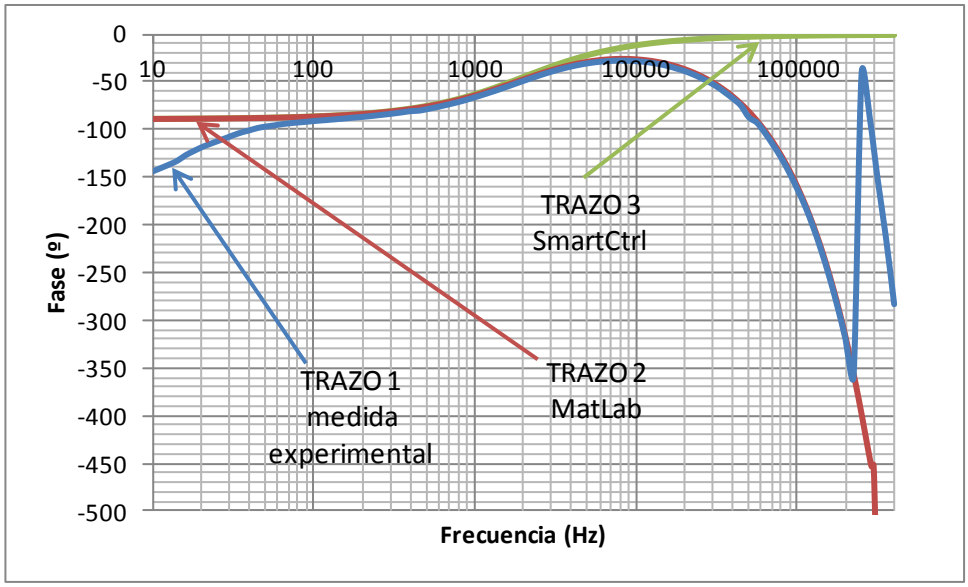


Figura 8-14 Resultados de fase del regulador PI_1

La medida de ganancia experimental del regulador coincide perfectamente con el regulador discreto de MatLab excepto para frecuencias muy bajas, pero desde 20Hz hasta 400kHz, coinciden perfectamente. Para la fase, los resultados coinciden desde 80Hz hasta 400kHz. Respecto al regulador de SmartCtrl, en ganancia también coincide perfectamente, pero en fase no por el efecto del retardo en los reguladores digitales, que son el medido y el de MatLab.

8.3.1.1.3 Lazo PI_1

Por último los resultados para el lazo abierto son:

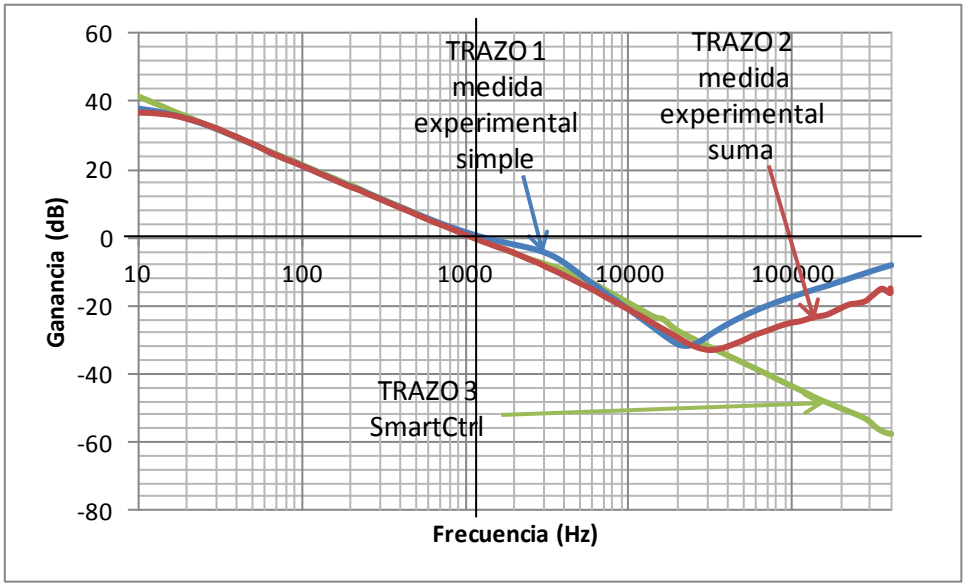


Figura 8-15 Resultados de ganancia de lazo abierto PI_1

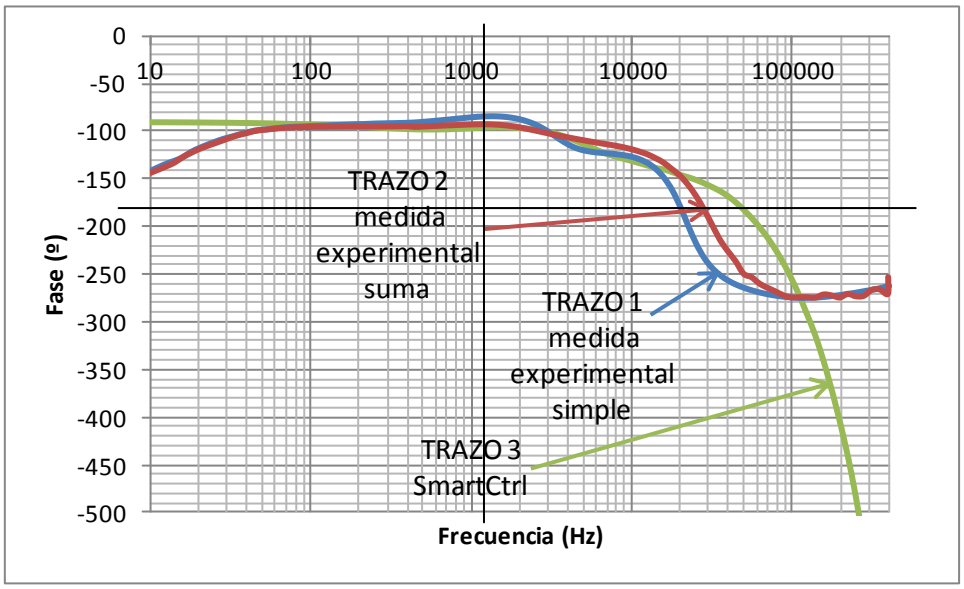


Figura 8-16 Resultados de la fase de lazo abierto PI_1

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	1,26	1,12	1,15478
MF (º)	180-84.82 = 95,18	180-93,42=86,58	84

Tabla 8-2 Resultados del PI 1

Los resultados teóricos de SmartCtrl son muy similares a los resultados de la suma de planta, regulador y sensor, y razonablemente similares a los medidos directamente. Sin embargo en los medidos directamente sobre el lazo, se ve una discrepancia con respecto a los anteriores en torno a 2000 y 4000Hz. Este problema no se ve en las medidas suma, que también son obtenidos experimentalmente.

En las dos medidas experimentales, se ve que a alta frecuencia difieren de lo teórico de SmartCtrl por efecto del comportamiento a altas frecuencias del sensor, que no ha sido modelado en el SmartCtrl.

8.3.1.2 PI_2

En el segundo caso la Fc está después de la frecuencia de resonancia del filtro de salida del convertidor y el margen de fase está dentro de los estándares del diseño de convertidores:

Frecuencia de cruce (Fc):	4,21697kHz
Margen de Fase (MF):	53º

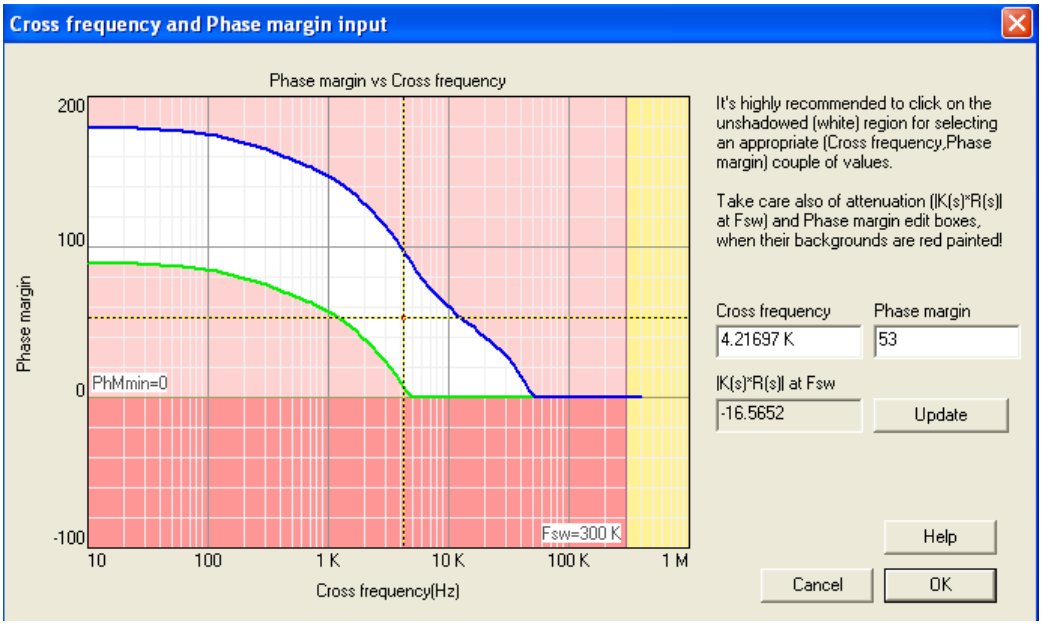


Figura 8-17 Figura 8-18 Mapa de soluciones PI_2

Para estas condiciones el sensor implementado es distinto al del caso anterior, en este caso el sensor fijado en SmartCtrl, que proviene de la propia ganancia del sensor más la del dsPIC, es la siguiente:

$$G_{dsPIC} \cdot G_{sensor} = \frac{1}{16} \times 32 \times \frac{1023}{3{,}3} \times \frac{1}{3205} \times \frac{5}{8{,}3} = 0{,}116533 \qquad \textit{Ec. 8-4}$$

Esta diferencia en el sensor de SmartCtrl, está provocada por la ganancia descrita en el apartado 7.1.2.2.4, y ha sido fijada mediante pruebas experimentales.

8.3.1.2.1 Planta PI_2

Los resultados de la planta han sido los siguientes:

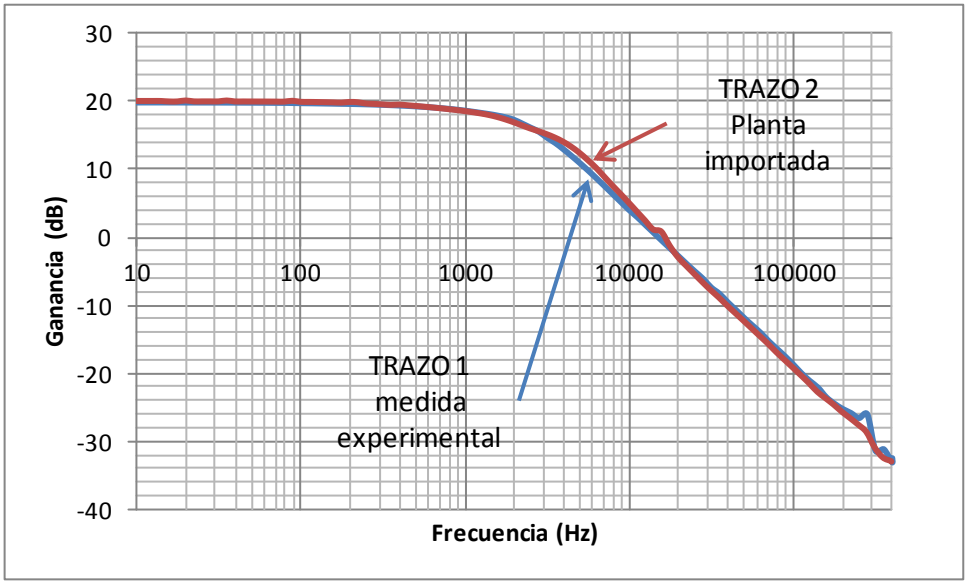


Figura 8-19 Resultados de la ganancia de la planta PI_2

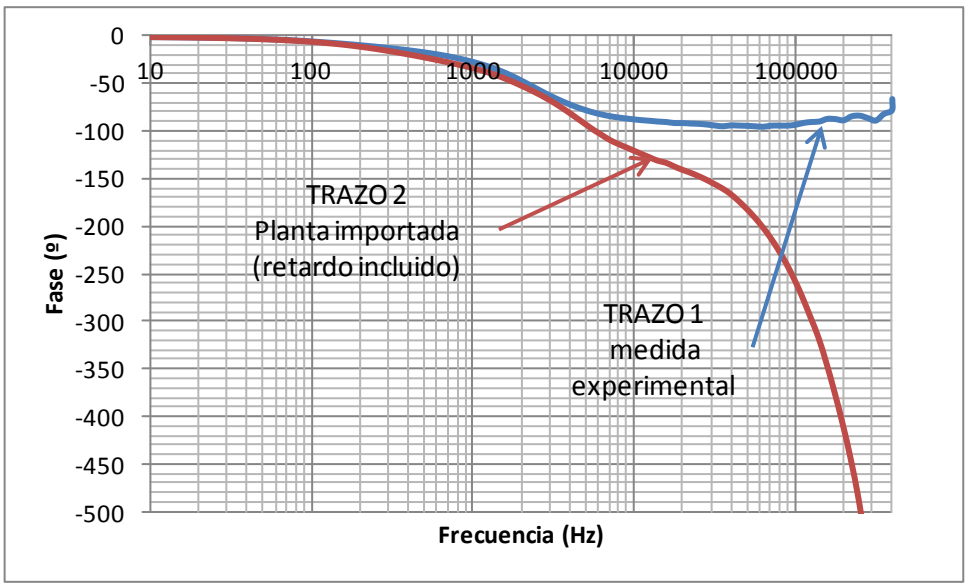


Figura 8-20 Resultados de la fase de la planta PI_2

De nuevo la planta importada y la experimental son coincidentes excepto en fase por efecto del retardo.

8.3.1.2.2 Regulador PI_2

Los resultados del regulador implementado en este segundo caso de PI, son los siguientes:

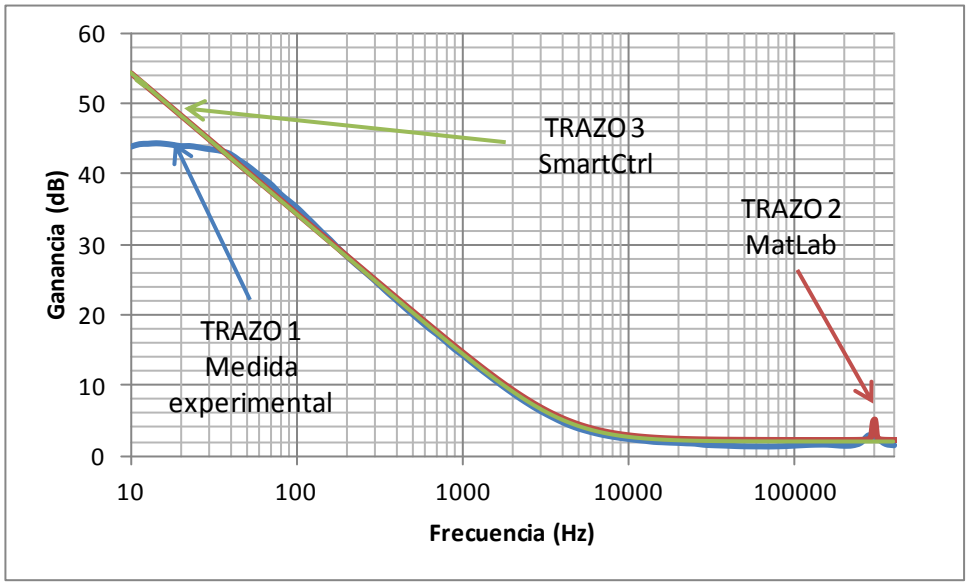


Figura 8-21 Resultados de ganancia del regulador PI_2

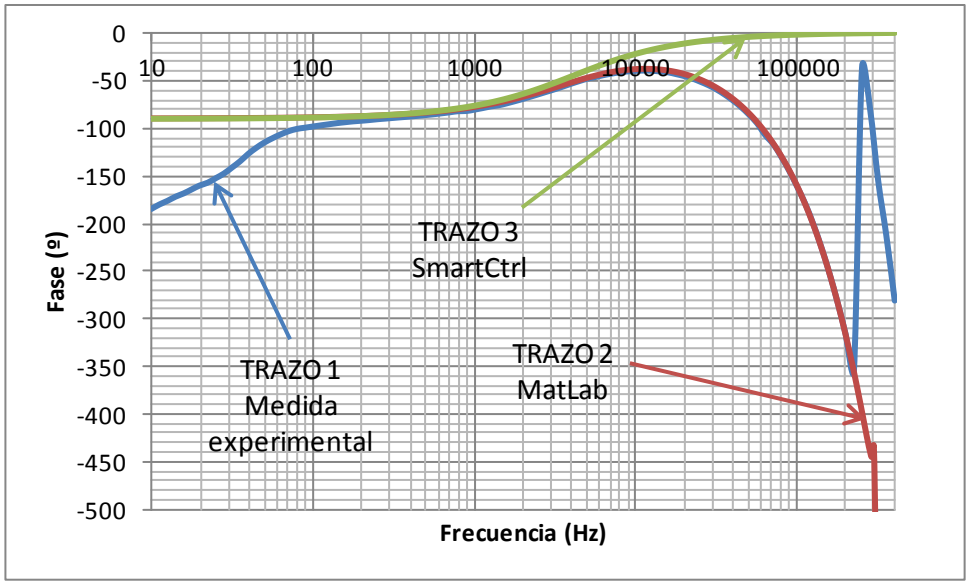


Figura 8-22 Resultados de fase del regulador PI_2

Como en el caso anterior, los resultados son muy positivos excepto para bajas frecuencias, precisamente cuando la ganancia del regulador es más alta. Sin embargo a partir de 40 Hz, que es una frecuencia relativamente baja, en ganancia los resultados son muy buenos. Para la fase los resultados empiezan a ser coincidentes con los teóricos de MatLab y SmartCtrl a partir de 80-90Hz. También en la fase se puede ver de nuevo la diferencia entre los reguladores digitales, el medido y el de MatLab, y el analógico de SmartCtrl.

8.3.1.2.3 Lazo PI_2

Por último para el lazo abierto los resultados son:

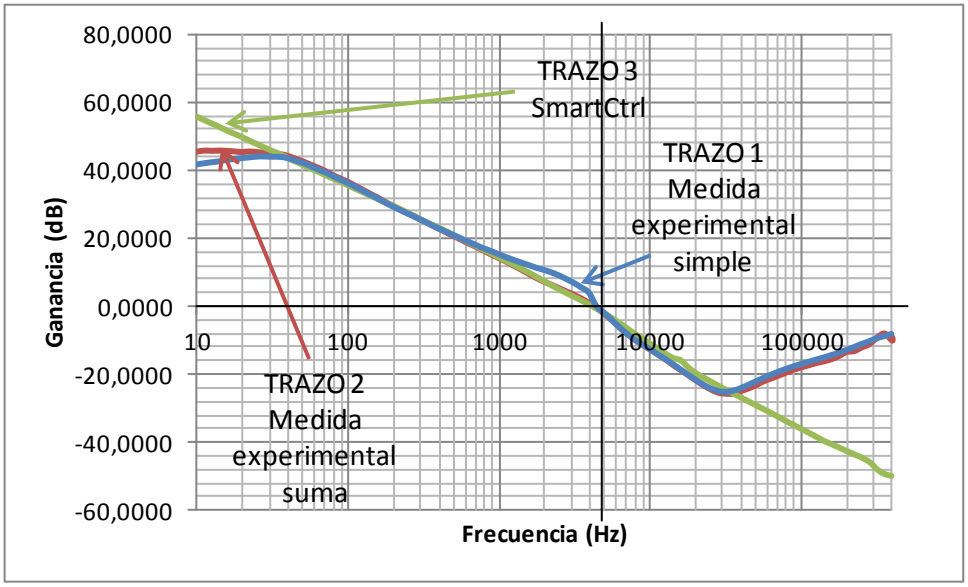


Figura 8-23 Resultados de ganancia de lazo abierto PI_2

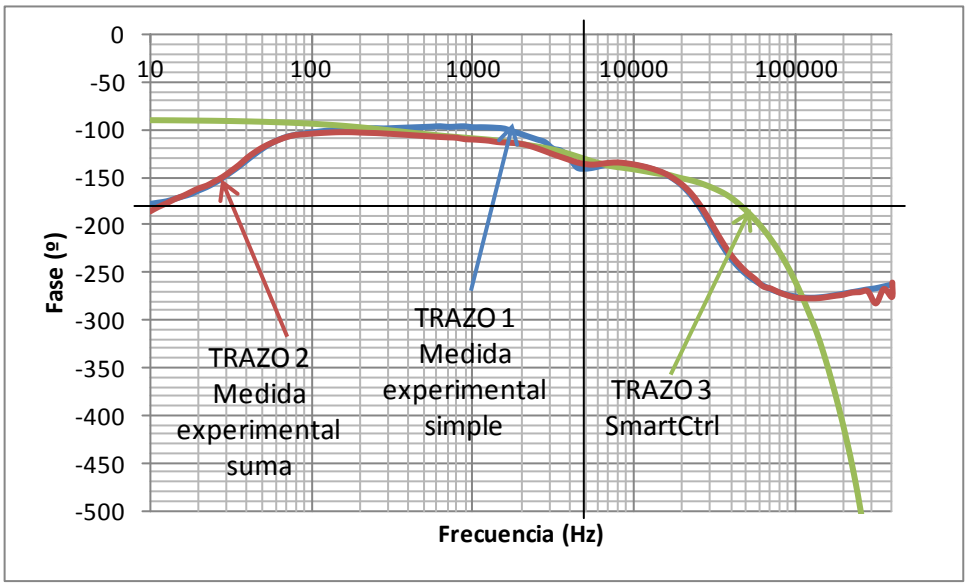


Figura 8-24 Resultados de fase de lazo abierto PI_2

Donde en las medidas experimentales simples son:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	4,46	4,2	4,21697
MF (º)	180-139,6 = 40,4	180-133 = 47	53

Tabla 8-3 Resultados del PI 2

Como en el caso PI_1, resultan mejores los resultados experimentales suma, que los simples. Si se observa de nuevo la planta y regulador medidos experimentalmente, se comprueba que la coincidencia de estos con lo teórico es muy buena, por lo que es lógico que la suma de ellos, más el sensor, de un resultado bueno. Esto es así excepto para alta frecuencia, precisamente por el comportamiento del sensor en altas, que es un fenómeno que no se tuvo en cuenta en el SmartCtrl al hacer el proceso de diseño descrito en el apartado 5. (Ver Figura 5-4 y Figura 5-5)

Por otro lado en la medida simple se ve como la ganancia y la fase, se despegan ligeramente de lo marcado teóricamente por SmartCtrl en torno a 2000-4000kHz, zona muy próxima a la de cruce por 0 dB. Sin embargo para el resto de frecuencias bajas y medias (hasta 20kHz, antes de los efectos de alta frecuencia del sensor) la medida directa sobre el lazo, resulta favorable y coincidente con la medida teórica que da SmartCtrl. Estos problemas y discrepancias también se observaron de forma muy similar en el PI_1.

8.3.1.3 PI_3

En este último caso de los PI, se ha fijado una Fc alta y el menor MF de los tres PI, con lo que el lazo implementado tiene la siguiente frecuencia de cruce y margen de fase:

Frecuencia de cruce (Fc):	10,227kHz
Margen de Fase (MF):	39,276º

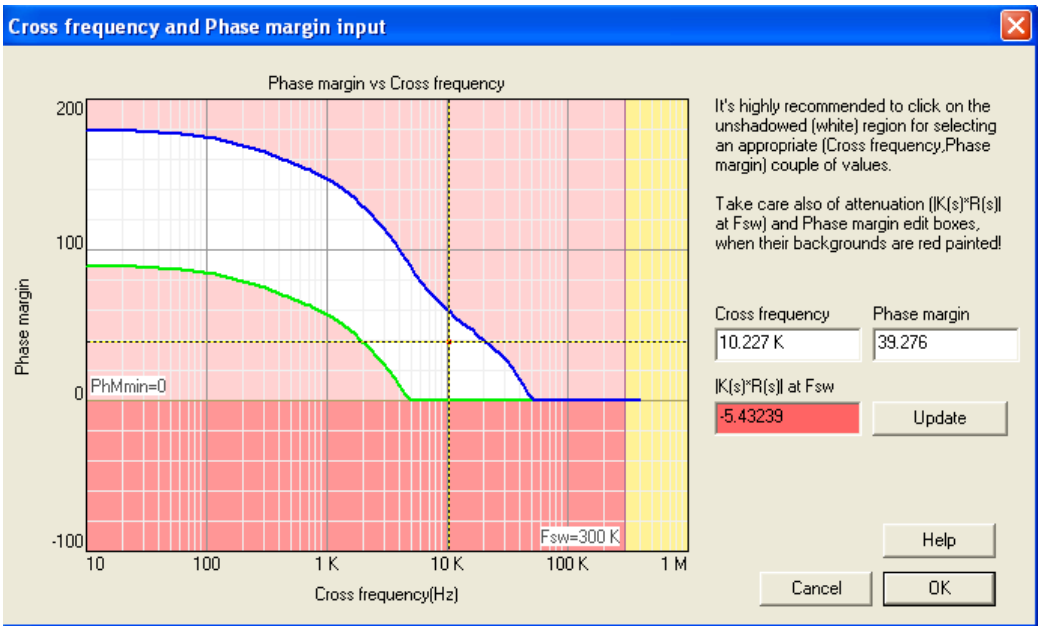


Figura 8-25 Mapa de soluciones PI_3

Para estas condiciones el sensor implementado en el SmartCtrl es el mismo que el del caso 8.3.1.1 y distinto del 8.3.1.2. Pero como en todos los casos, el sensor fijado proviene de la ganancia del sensor más la del dsPIC, y en este caso es la siguiente:

$$G_{dsPIC} \cdot G_{sensor} = \frac{1}{8} \times 32 \times \frac{1023}{3{,}3} \times \frac{1}{3205} \times \frac{5}{8{,}3} = 0{,}23307 \qquad \textit{Ec. 8-5}$$

De nuevo se recuerda que esta diferencia en el sensor está provocada por la ganancia introducida tal y como indica el apartado 7.1.2.2.6 y que es fijada experimentalmente para cada caso.

8.3.1.3.1 Planta PI_3

Los resultados de la planta han sido los siguientes:

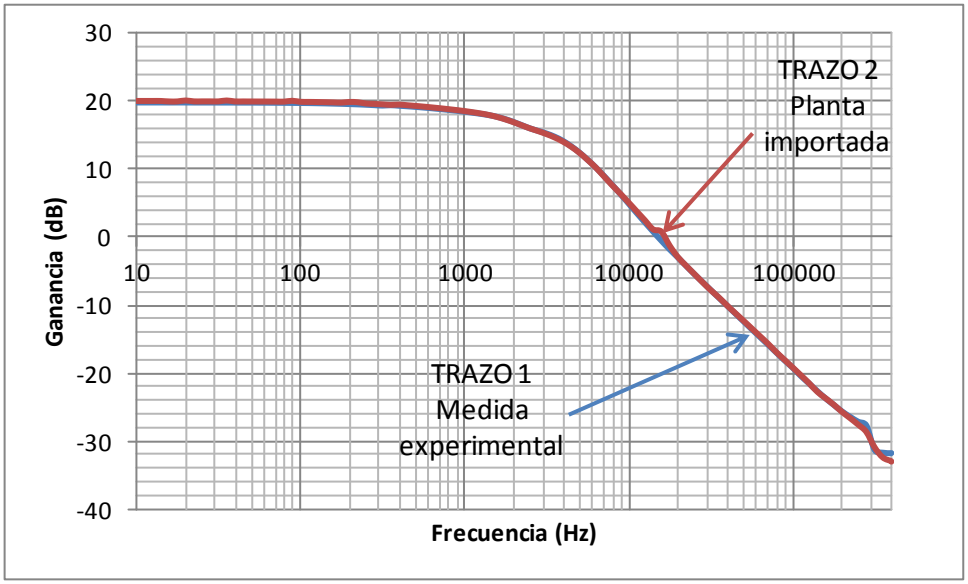


Figura 8-26 Resultados de la ganancia de la planta PI_3

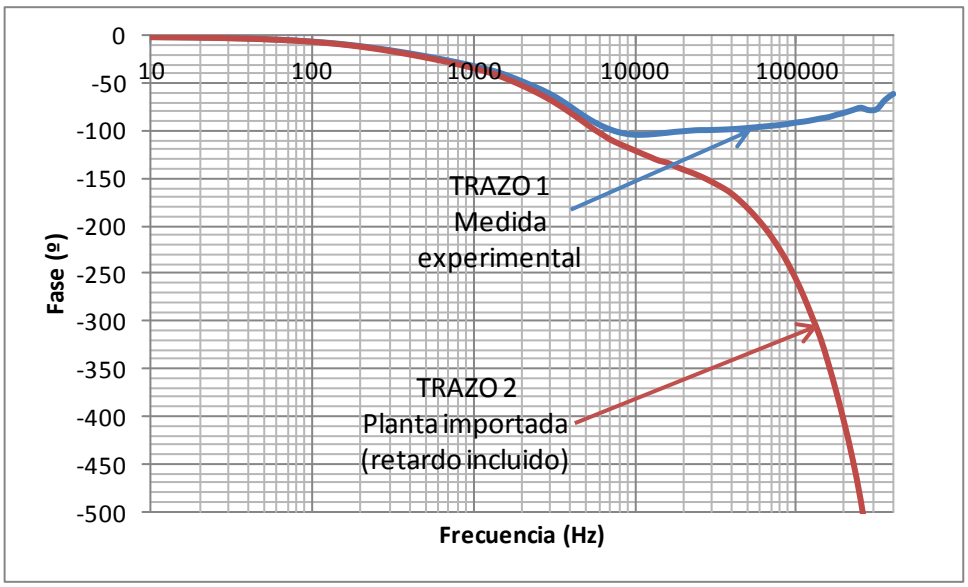


Figura 8-27 Resultados de la fase de la planta PI_3

Planta importada y experimental son coincidentes excepto en fase por efecto del retardo que es mayor cuanto mayor es la frecuencia.

8.3.1.3.2 Regulador PI_3

Los resultados del regulador implementado en este último caso de PI, son los siguientes:

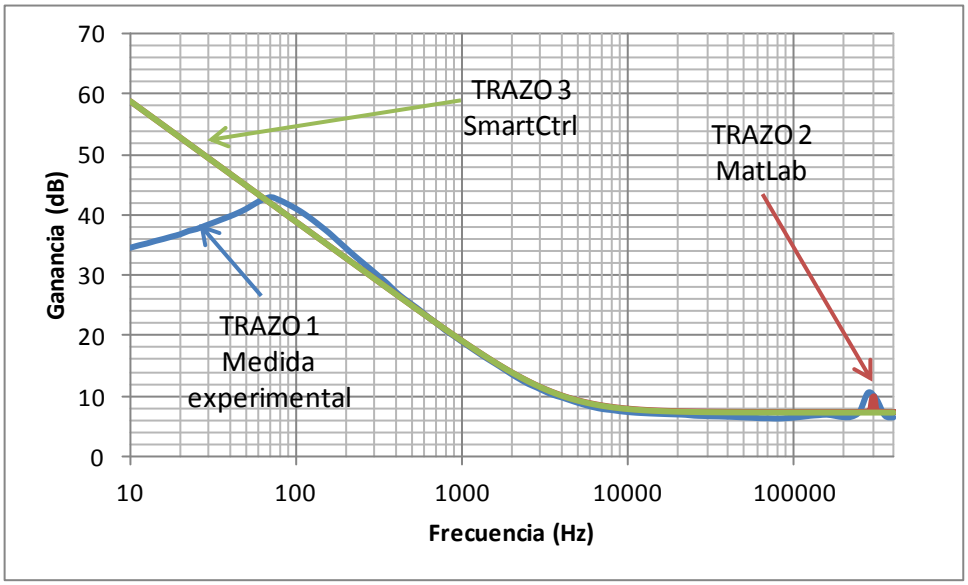


Figura 8-28 Resultados de la ganancia del regulador PI_3

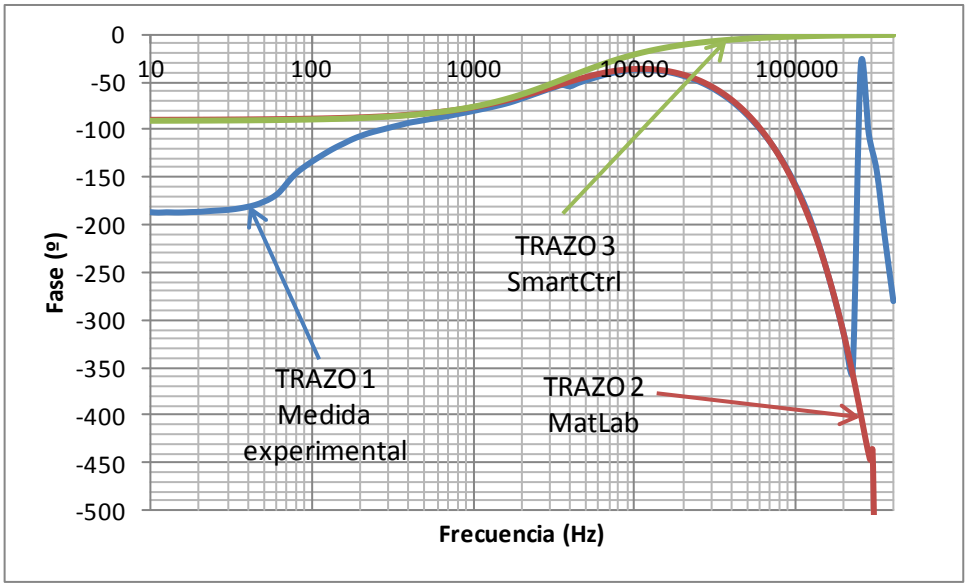


Figura 8-29 Resultados de la fase del regulador PI_3

Este tercer caso es el que da peores resultados a bajas frecuencias de los tres PI mostrados. El efecto a bajas frecuencias es muy similar al de los casos anteriores, pero en este caso, el rango de frecuencias donde no coincide lo medido con lo teórico es mayor, llegando a 70 Hz para la ganancia, pero hasta más de 500 Hz en la fase. La principal diferencia con los anteriores, es que este regulador en teoría debe arrancar con una ganancia mayor que en los casos anteriores, y como se comentó en el apartado 8.1, al tener el regulador un polo en el origen, la ganancia en bajas frecuencias es muy alta. Esto a efectos de medida supone comparar dos medidas en las cuales una de ellas está muy atenuada con respecto a la otra, lo

que provoca que sea muy difícil comparar correctamente ambas medidas tanto en ganancia como en fase (Ver Figura 8-2). Este fenómeno provoca que a bajas frecuencias, es decir cuando la ganancia de la medida es mayor, los resultados medidos no estén bien, y no coincidan con lo esperado teóricamente de MatLab y SmartCtrl.

Por otro lado para frecuencias medias y altas la coincidencia entre lo teórico y lo medido es muy alta, más aún en el caso del regulador de MatLab que es un regulador discreto, donde el de SmartCtrl es distinto, ya que este es continuo y no tiene el retardo de los discretos, el real y el de MatLab.

8.3.1.3.3 Lazo PI_3

Finalmente para el lazo los resultados son:

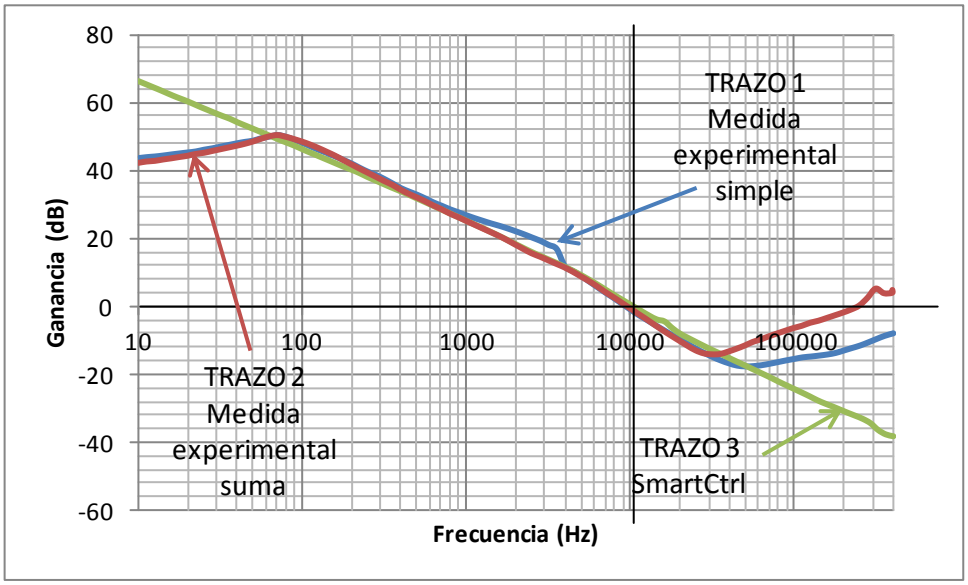


Figura 8-30 Resultados de ganancia del lazo abierto PI_3

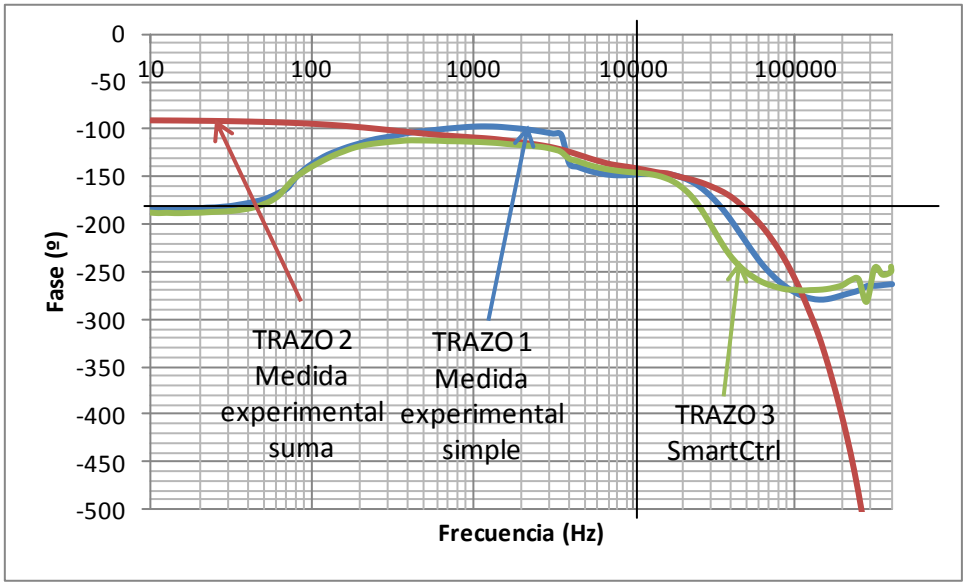


Figura 8-31 Resultados de fase del lazo abierto PI_3

Las medidas experimentales simples son:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	9,5	9,5	10,227
MF (º)	180-147,8 = 32,2	180-144,7 = 35,3	39,276

Tabla 8-4 Resultados del PI 3

En este último caso, los resultados experimentales suma resultan de nuevo mejores que los simples. En estos últimos se produce el mismo defecto que en el PI_1 y PI_2, es decir hay una discrepancia entre lo teórico de SmartCtrl y la medida simple en torno a 2000 y 4000Hz. De nuevo las medidas suma y las teóricas son muy parecidas, ya que planta y regulador medidos por separado coinciden con lo teórico, por lo que es de esperar que la suma de estos más el sensor de resultados muy parecidos a lo teórico de SmartCtrl. Esto es así hasta que predomina la influencia del sensor a altas frecuencias, que no fue contemplado en el diseño mediante SmartCtrl.

8.3.1.4 Conclusiones reguladores PI

El proceso de diseño desarrollado y usado para obtener estos reguladores en dsPIC, se considera válido ya que cumple razonablemente con lo diseñado teóricamente.

Se han dado algunas particularidades que se repiten en los tres ejemplos. Por un lado a bajas frecuencias, para la medida del regulador y del lazo, los cuales tienen polo en el origen, se ve que hay problemas al hacer la medida y esta no sale como debiera. Este problema es especialmente importante en el último ejemplo, ya que es el que mayor ganancia presenta en bajas frecuencias, y por efecto del sistema de medida, las señales con mucha ganancia tienen dificultades en ser medidas. (Figura 8-2)

Por otro lado al medir el lazo, además del problema anterior, a altas frecuencias se tiene el efecto provocado por el sensor. Este tiene un comportamiento a alta frecuencia que no ha sido contemplado ni incluido en el proceso de diseño con SmartCtrl, de ahí que lo experimental y lo teórico de SmartCtrl difieran.

Por último, en los tres casos al medir el lazo directamente, se ve que entre 2000Hz y 4000Hz, la medida se aleja ligeramente de lo que cabría esperar. Este fenómeno es provocado por el sistema de medida, ya que la misma medida hecha por bloques, no da ese problema.

Como se ha visto con los tres ejemplos anteriores, lo implementado en el dsPIC cumple razonablemente bien con que se diseño teóricamente con ayuda de SmartCtrl. Por lo tanto ha quedado validado experimentalmente que es posible diseñar reguladores PI mediante el SmartCtrl, y luego implementarlos de una manera sencilla en un dsPIC para controlar convertidores comerciales.

8.3.2 Reguladores tipo 2

En este grupo están los tres reguladores de tipo 2, cuyas FDT tienen la forma:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_1 s+1}{a_2 s^2+a_1 s} \qquad \textit{Ec. 8-6}$$

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \textit{Ec. 8-7}$$

Los márgenes de fase y frecuencia de corte seleccionados para este tipo de control, son los que se ven en la Figura 8-32.

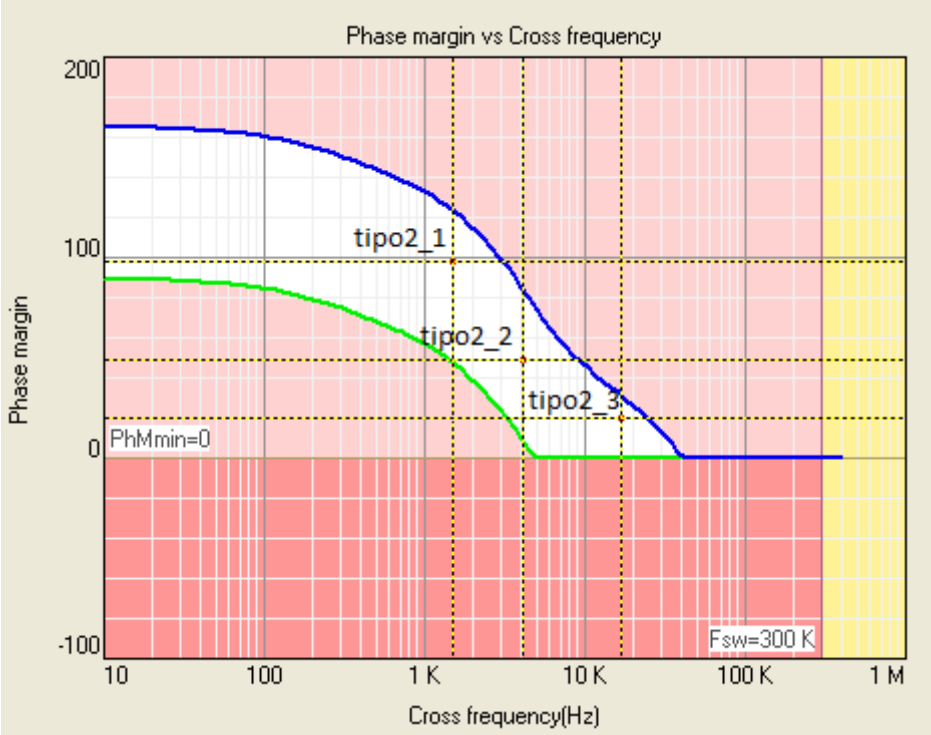


Figura 8-32 Mapa de soluciones para controles tipo 2

En este caso se ve el mapa de soluciones con los tres ejemplos de reguladores tipo2, implementados en este proyecto para justificar el funcionamiento del proceso de diseño para este tipo de regulador. Como en el caso anterior se han fijado puntos separados para cubrir una mayor área del mapa de soluciones.

Los valores de frecuencia de cruce y margen de fase fijados como teóricos con el SmartCtrl para estos tres lazos son los siguientes:

	tipo 2_1	tipo 2_2	tipo 2_3
Fc (kHz)	1,48653	4,09732	16,6511
MF (º)	98,172	49	19,62

Tabla 8-5 Frecuencia de cruce (fc) y margen de fase (MF) para lazos de tipo 2

A diferencia del caso de los PI, los ejemplos del tipo2, tienen todos la misma ganancia del sensor en SmartCtrl que se introdujo en el apartado 7.1.2.2.6 y que responde a la siguiente expresión:

$$G_{dsPIC} \cdot G_{sensor} = \frac{1}{4} \times 32 \times \frac{1023}{3{,}3} \times \frac{1}{3205} \times \frac{5}{8{,}3} = 0{,}464333 \qquad \textit{Ec. 8-8}$$

8.3.2.1 Tipo2_1

En este primer caso de regulador tipo2 se tiene un control que fija la Fc antes de la frecuencia de resonancia del filtro de salida del convertidor, con un margen de fase muy alto:

Frecuencia de cruce (Fc):	1,48653kHz
Margen de Fase (MF):	98.172º

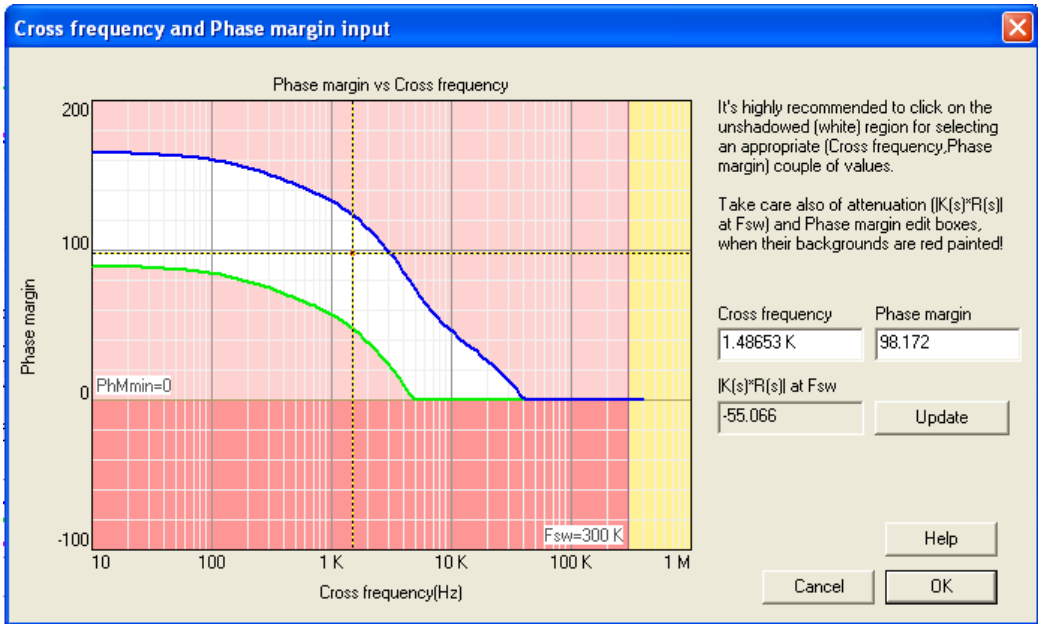


Figura 8-33 Mapa de soluciones tipo2_1

8.3.2.1.1 Planta tipo2_1

Los resultados de la planta han sido los siguientes:

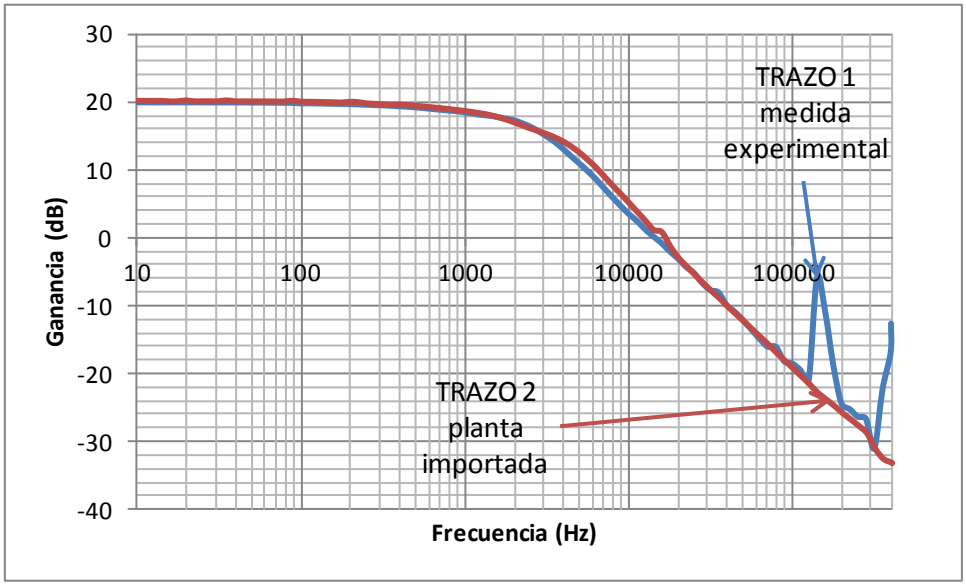


Figura 8-34 Resultados de ganancia de la planta tipo2_1

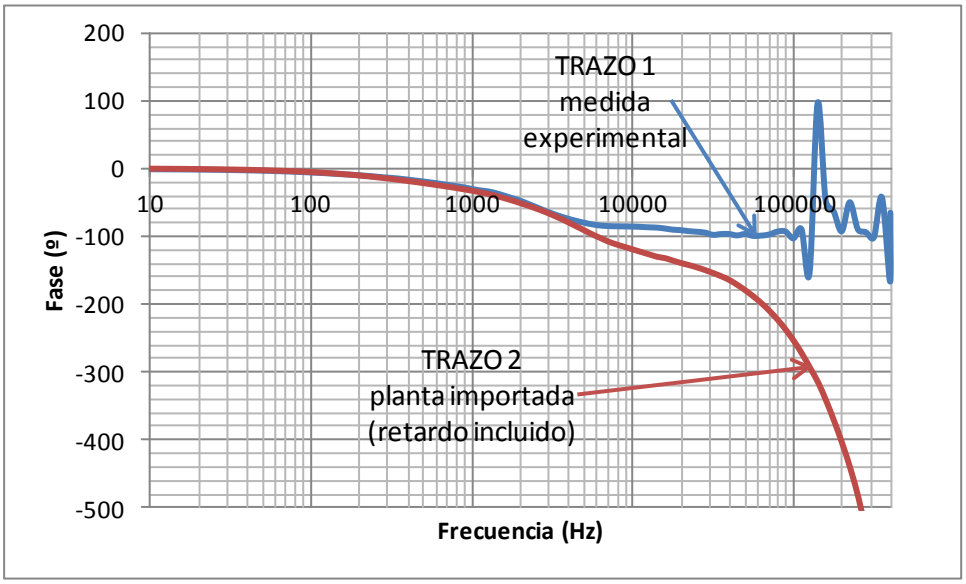


Figura 8-35 Resultados de fase de la planta tipo2_1

Con carácter general los resultados obtenidos son coincidentes en ganancia excepto una medida en torno a 140kHz. Este fallo se debe a un error en la media ya que de los datos de la misma gráfica se puede ver que es sólo un dato y sus adyacentes siguen con la pendiente normal. Para la fase se ve que la principal discrepancia es la producida por el retardo, además de un pico a la misma frecuencia que en la ganancia donde se vio un error de medida. Finalmente se tiene que para bajas y medias frecuencias, las dos medidas son bastante coincidentes.

8.3.2.1.2 Regulador tipo2_1

Los resultados del primer regulador implementado de tipo2, son los siguientes:

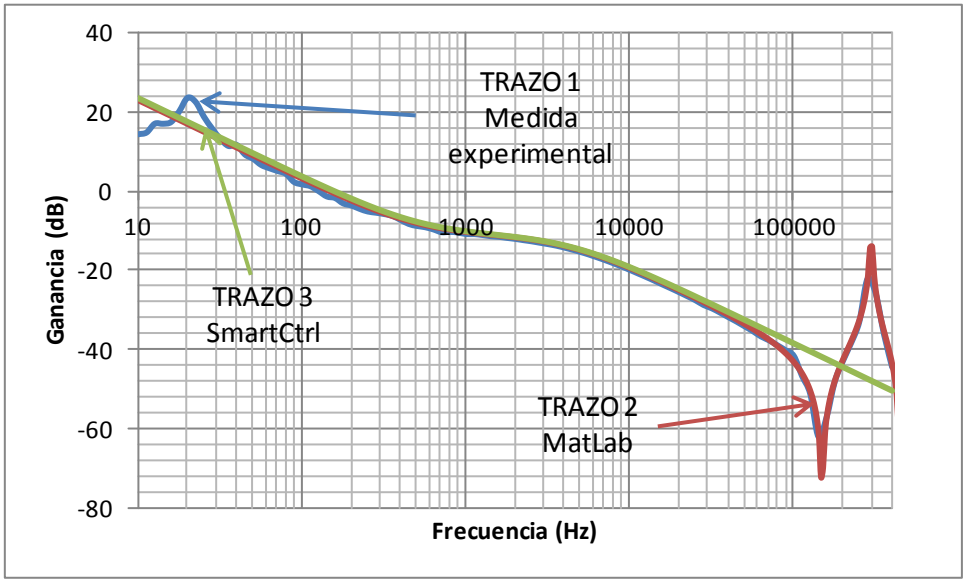


Figura 8-36 Resultados de ganancia del regulador tipo2_1

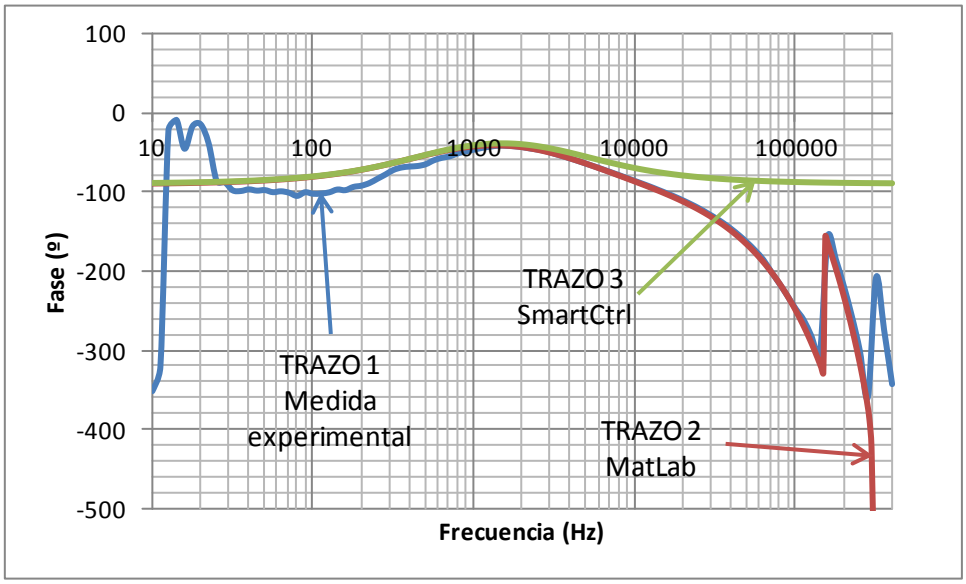


Figura 8-37 Resultados de fase del regulador tipo2_1

La medida de ganancia es muy satisfactoria por su coincidencia clara con lo esperado e indicado por SmartCtrl y MatLab. Sólo existe una pequeña discrepancia en bajas frecuencias precisamente cuando mayor es la ganancia, pero para el resto de frecuencias la coincidencia es total. Con respecto al de SmartCtrl, en altas frecuencias se separan por el efecto de la frecuencia de conmutación, pero con regulador discreto de MatLab la coincidencia es evidente.

Para la fase ocurre algo parecido, sin embargo si hay una pequeña diferencia entre lo medido y lo teórico para frecuencias hasta 300Hz, a partir de ahí la coincidencia con el trazo 2, el discreto, es muy satisfactoria.

8.3.2.1.3 Lazo tipo2_1

Por último los resultados para el lazo abierto son:

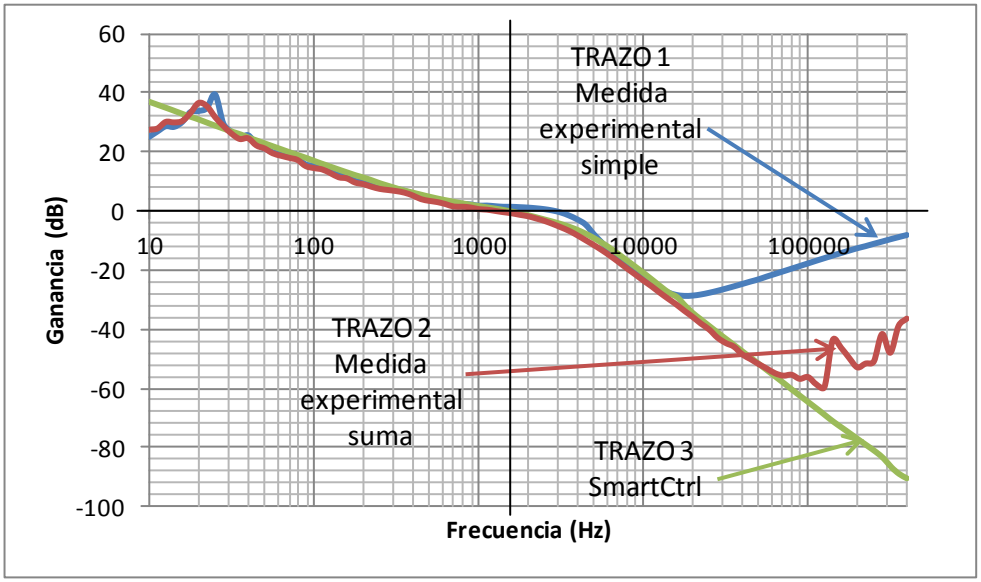


Figura 8-38 Resultados de ganancia del lazo abierto tipo2_1

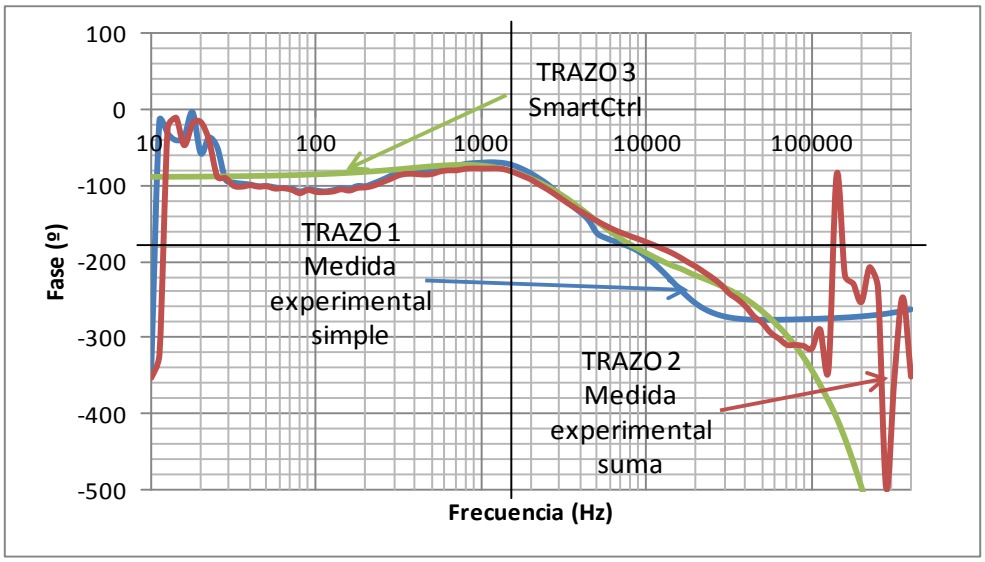


Figura 8-39 Resultados de fase del lazo abierto tipo2_1

Donde en las medidas experimentales simples:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	2,8	1,413	1,48653
MF (º)	180-107,6 = 72,4 MF(1,5kHz)=105	180-78,8 = 101,2	98,172

Tabla 8-6 Resultados del tipo2_1

Por lo tanto para este primer regulador de tipo2, las medidas hechas por separado de planta, regulador y sensor sumados, dan unos resultados muy parecidos a lo esperado y marcado por SmartCtrl. Sin embargo para las medidas tomadas directamente sobre el lazo, se tienen los mismos problemas que en el caso de los PI pero en este caso para un rango mayor de frecuencias, desde 1kHz hasta 5kHz, con lo que la ganancia y fase medidas se alejan de lo teórico. Esta separación a nivel de las gráficas puede apreciarse y no es muy grande, sin embargo en este caso afecta precisamente a la zona de la frecuencia de cruce, con lo que Fc y MF quedan algo alejados de lo esperado. Por este motivo se muestran dos casos de margen de fase, uno a la frecuencia de cruce medida y otro con la frecuencia de cruce teórica.

Sin embargo se insiste en que las medias de planta y regulador han sido muy similares a lo esperado, de ahí que la suma de ambas con el sensor, haya dado unos resultados muy buenos por su precisión en Fc y MF respecto de lo esperado.

8.3.2.2 Tipo2_2

En el segundo caso de regulador tipo2, se tiene un control que se considera dentro de los tres, el ejemplo más razonable por su Fc y MF seleccionado:

Frecuencia de cruce (Fc):	4,09732kHz
Margen de Fase (MF):	49º

Este regulador Tipo2_2 presenta una particularidad que ha sido expresamente introducida para mostrar una particularidad del diseño. Como se comenta en el proceso de diseño del apartado 5.4, en el paso 10 de la Tabla 5-1, puede darse el caso de que los coeficientes introducidos en el programa, no cumplan la función de integrador por una mala interpretación de los coeficientes al pasarlos a Q15. En este caso, la función de integrador no se cumple perfectamente. Con un integrador ideal (polo en el origen) la pendiente de bajada de la ganancia desde 0 Hz es de -20dB/década, lo que con estas medidas de ganancia es difícil de apreciar ya que en bajas frecuencias y grandes ganancias hay errores en la medida. Sin embargo para un integrador ideal la fase inicial, a 0 Hz, debe ser de -90º, y es en este efecto donde se puede ver mejor en los resultados experimentales que pasa cuando el control no tiene integrador ideal, al no haber revisado los coeficientes una vez traducidos al formato Q15.

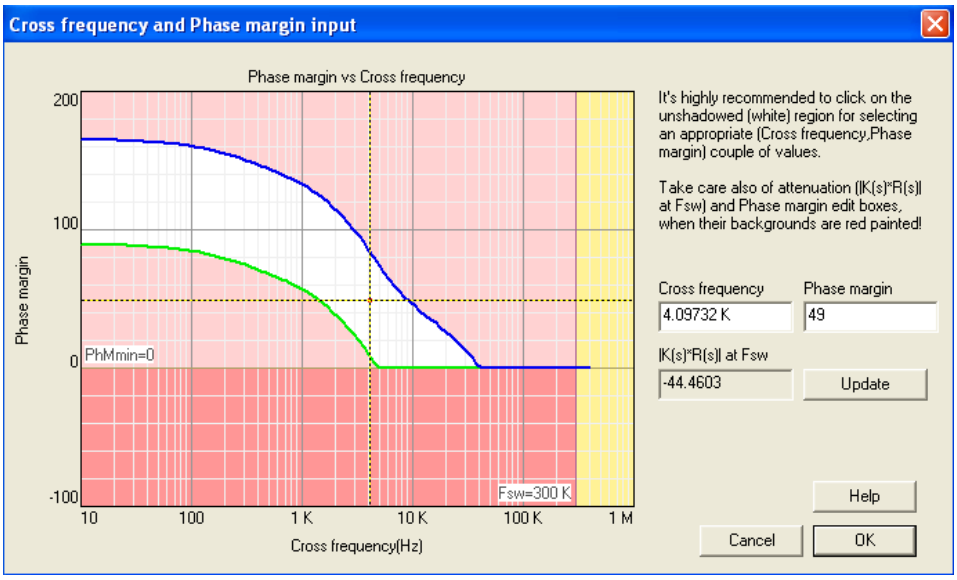


Figura 8-40 Mapa de soluciones tipo2_2

8.3.2.2.1 Planta tipo2_2

Los resultados de la planta han sido los siguientes:

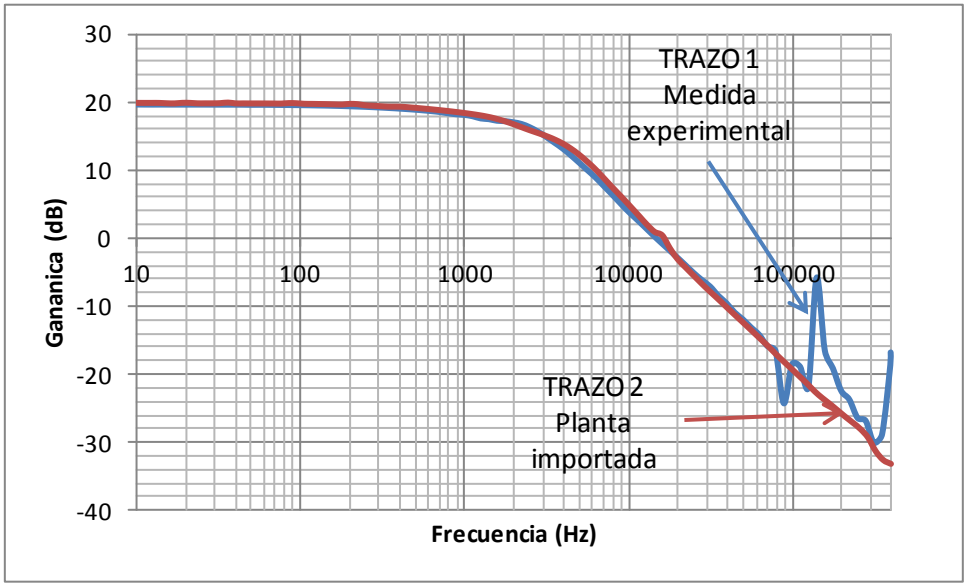


Figura 8-41 Resultados de ganancia de la planta tipo2_2

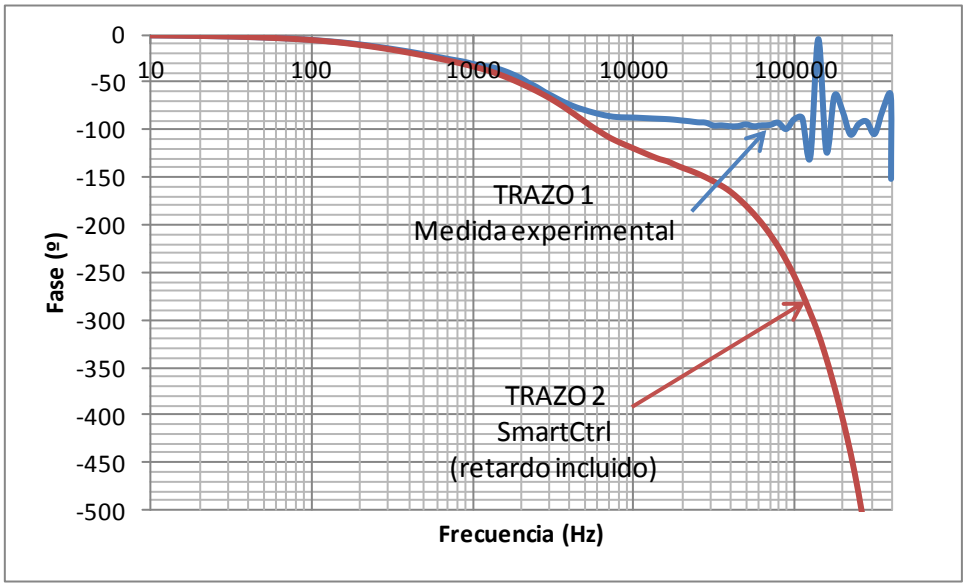


Figura 8-42 Resultados de fase de la planta tipo2_2

De nuevo, con carácter general la planta medida y la que fue importada a SmartCtrl para hacer el diseño del lazo, son bastante parecidas excepto a frecuencias próximas a 100kHz o superiores, que son frecuencias ya cercanas a la frecuencia de conmutación. En cualquier caso en la fase sigue predominando el efecto del retardo introducido en la planta importada para que este fuese tenido en cuenta en el diseño mediante SmartCtrl.

8.3.2.2.2 Regulador tipo2_2

Para este segundo regulador tipo2 sin integrador ideal, los resultados experimentales han sido los siguientes:

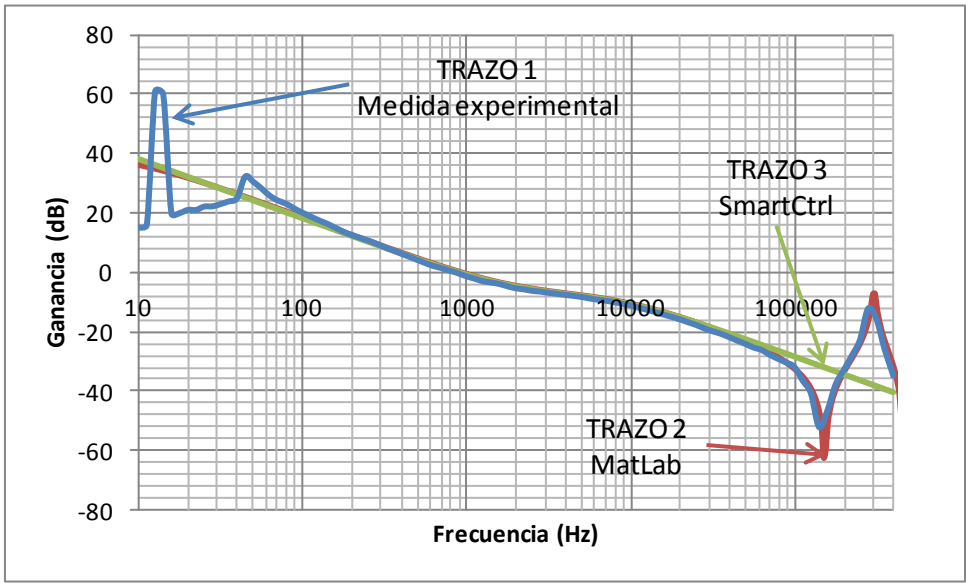


Figura 8-43 Resultados de ganancia del regulador tipo2_2

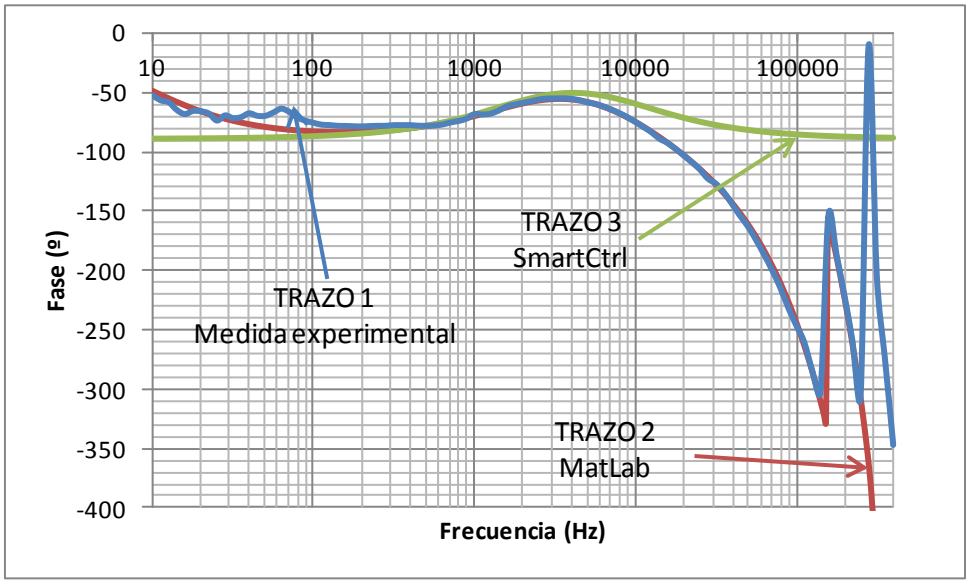


Figura 8-44 Resultados de fase del regulador tipo2_2

Como puede verse, los resultados experimentales se ajustan muy bien a lo teórico del SmartCtrl y mejor aún a lo de MatLab en un amplio rango de frecuencia. Esto es así porque además de que los dos reguladores son digitales, el de MatLab tiene exactamente los mismos coeficientes que se han implementado en el experimental, donde ninguno cumple la función de integrador. En cambio el de SmartCtrl sí, ya que es un tipo2 y de por sí, tiene polo un el origen. Sin embargo en el real, en el proceso de conversión a Q15, esta conversión introduce un error de resolución que sí se tiene en cuenta en MatLab, ya que en él, el usuario puede

introducir los coeficientes que quiera, en este caso los mismos que interpreta el dsPIC al pasar números reales a Q15.

Para bajas frecuencias, en ganancia se tienen los mismos problemas de medida que en casos anteriores y no puede apreciarse mucho este efecto. Sin embargo en la fase, mientras la de SmartCtrl empieza en -90º como integrador ideal, en MatLab y en lo medido se ve que no es así, sino que viene desde 0º y va a -90º por efecto de un polo próximo al origen, pero no en él.Esto es así porque los coeficientes no cumplen la condición descrita en 6.2.5.5, donde el sumatorio de los coeficientes del denominador debe ser 0. En este caso, ese sumatorio no es 0, pero sí muy próximo, tanto como un valor más que 0 en Q15, es decir:

$$Suma\ de\ coeficientes = 2^{-15} = 3.0517578 \cdot 10^{-5} \qquad Ec.\ 8\text{-}9$$

Para terminar, se puede decir que el comportamiento de la medida experimental, se ajusta muy bien a lo esperado tanto para ganancia como para fase en un amplio rango de frecuencia como se ve en la Figura 8-43 y Figura 8-44.

8.3.2.2.3 Lazo tipo2_2

Por último los resultados para el lazo son:

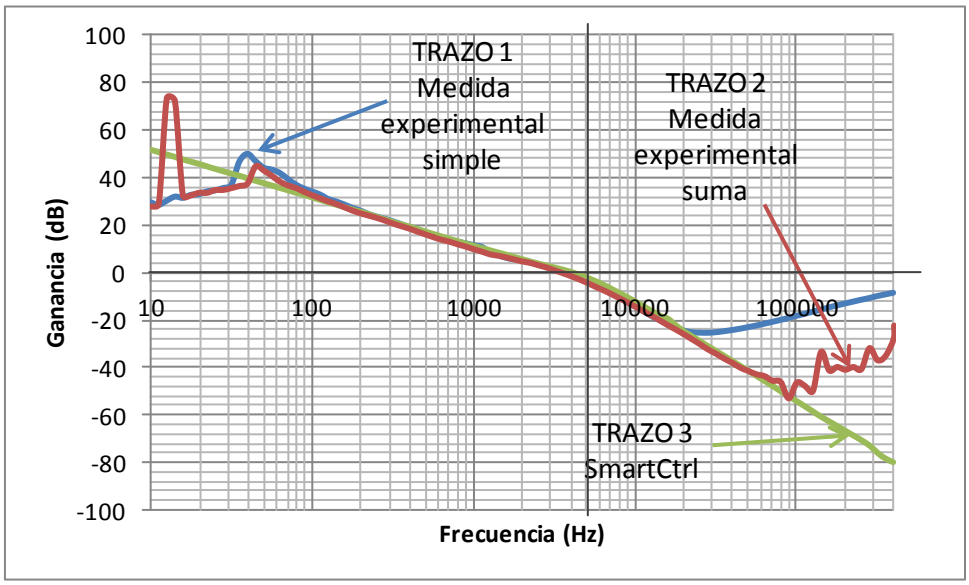


Figura 8-45 Resultados de ganancia del lazo abierto tipo2_2

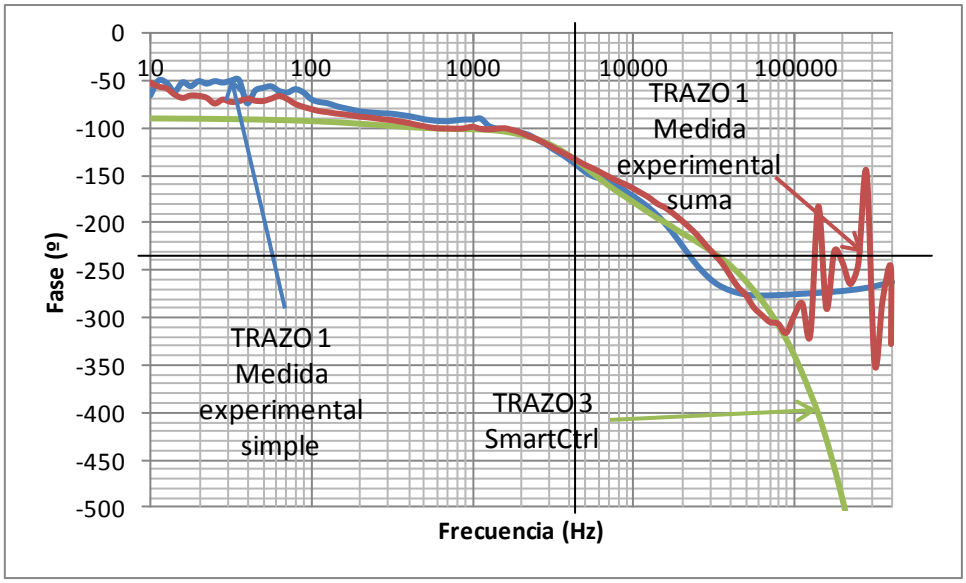


Figura 8-46 Resultados de fase del lazo abierto tipo2_2

De las medidas experimentales simples se tiene que:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	4,05	3,65	4,09732
MF (º)	180-134,8 = 45,2	180-125,9 = 54,1	49

Tabla 8-7 Resultados del tipo2_2

En este caso los resultados experimentales simples se ajustan más a lo teórico, especialmente en frecuencia de cruce. El valor experimental suma de regulador, planta y sensor queda en la gráfica ligeramente por debajo. Esto es debido a que precisamente tanto el regulador como la planta medidos y mostrados en los apartados anteriores, son ligeramente inferiores de lo teórico aunque en los gráficos casi no se pueda apreciar. Esos dos pequeños errores, una vez sumados se hacen un poco más significativos y aunque de nuevo en la gráfica casi no se aprecia, si provoca que la frecuencia de cruce esté en torno a 450Hz menos de lo que debería. Para la fase la medida experimental simple y suma son bastante parecidas. Como hemos visto en este ejemplo, no se tiene la función de integrador ideal, de ahí que a bajas frecuencias el desfase no empiece en -90º, como sí lo hace el teórico que si tiene integrador ideal. Por eso mismo para bajas frecuencias, lo medido y el resultado de SmartCtrl es distinto, ya que el teórico tiene integrador ideal y lo medido no. Para frecuencias medias las medidas están razonablemente bien y para altas, la influencia del sensor hace que lo medido se aleje de lo teórico de SmartCtrl que no contempla el efecto del sensor a alta frecuencia.

En cualquier caso los resultados mostrados en las figuras anteriores son razonablemente buenos y se ajustan bien a lo teórico esperado para un amplio rango de frecuencia.

8.3.2.3 Tipo2_3

En el último lazo de control de tipo2, se ha buscado una frecuencia de cruce alta y margen un de fase bajo, cuyas características exactas son:

Frecuencia de cruce (Fc):	16,6511kHz
Margen de Fase (MF):	19,62º

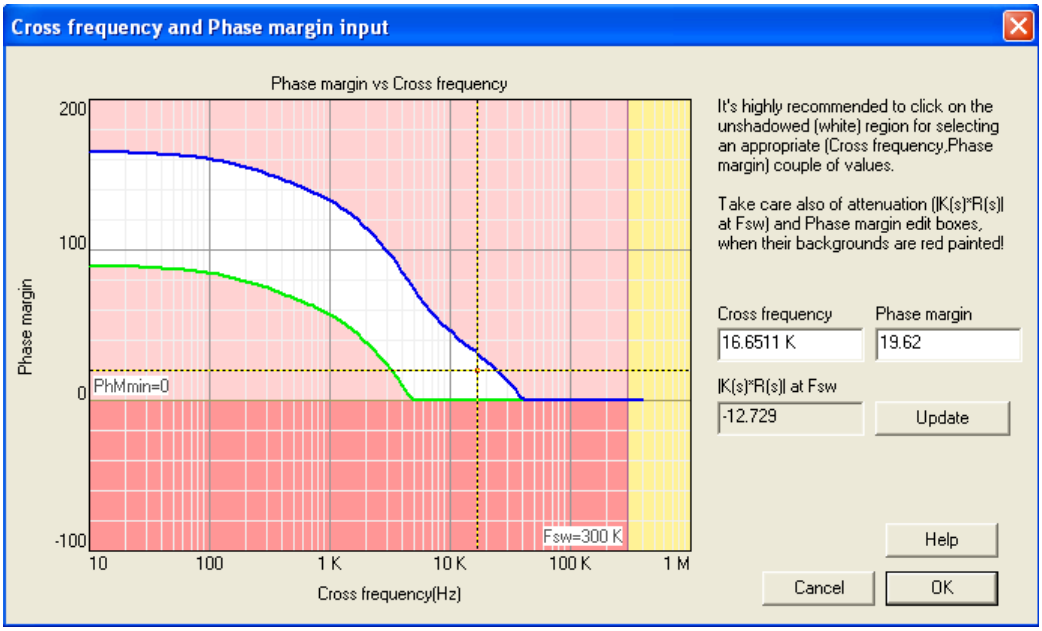


Figura 8-47 Mapa de soluciones tipo2_3

8.3.2.3.1 Planta tipo2_3

Los resultados de la planta han sido los siguientes:

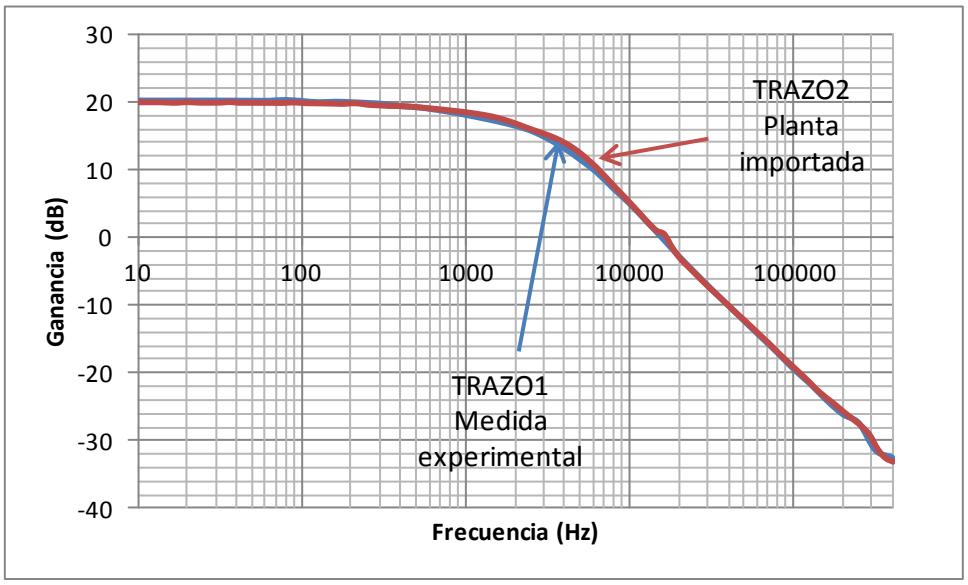


Figura 8-48 Resultados de ganancia de la planta tipo2_3

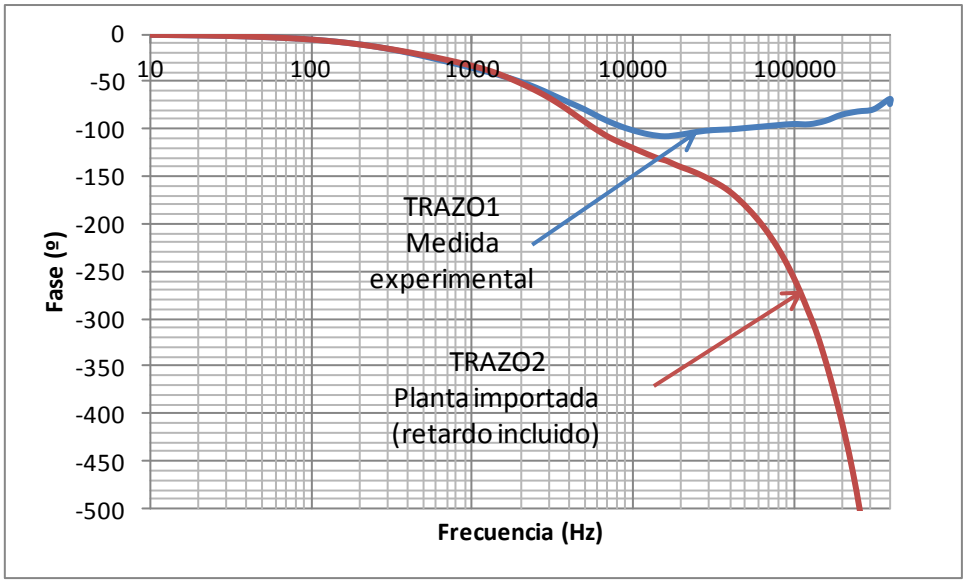


Figura 8-49 Resultados de fase de la planta tipo2_3

Como en todos los casos anteriores, en ganancia, la planta importada y la medida para este caso concreto, son muy parecidas. Para la fase, a frecuencias bajas también son muy parecidas hasta que la influencia del retardo, se empieza a hacer muy significativa a medida que la frecuencia aumenta.

8.3.2.3.2 Regulador tipo2_3

En este último regulador tipo2, los resultados experimentales han sido los siguientes:

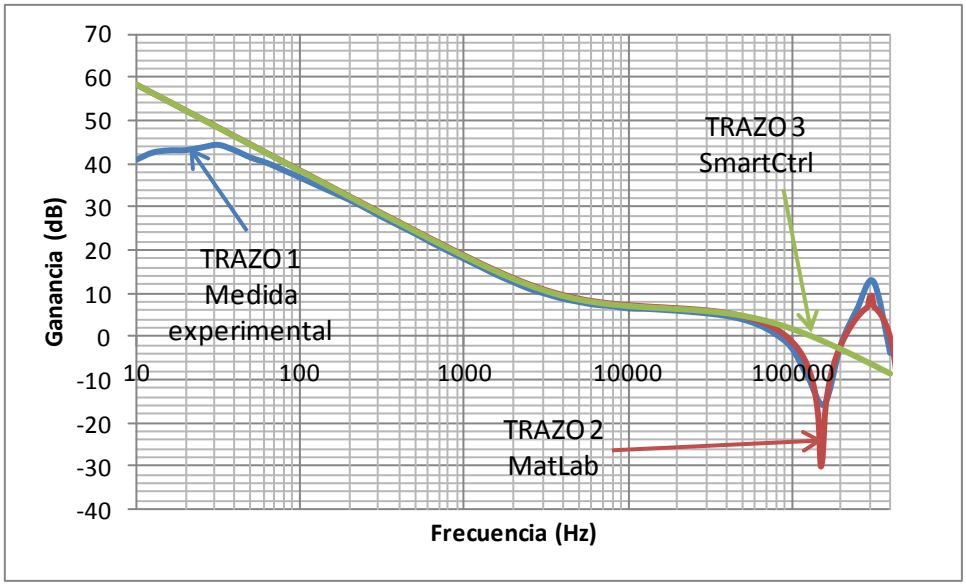


Figura 8-50 Resultados de ganancia del regulador tipo2_3

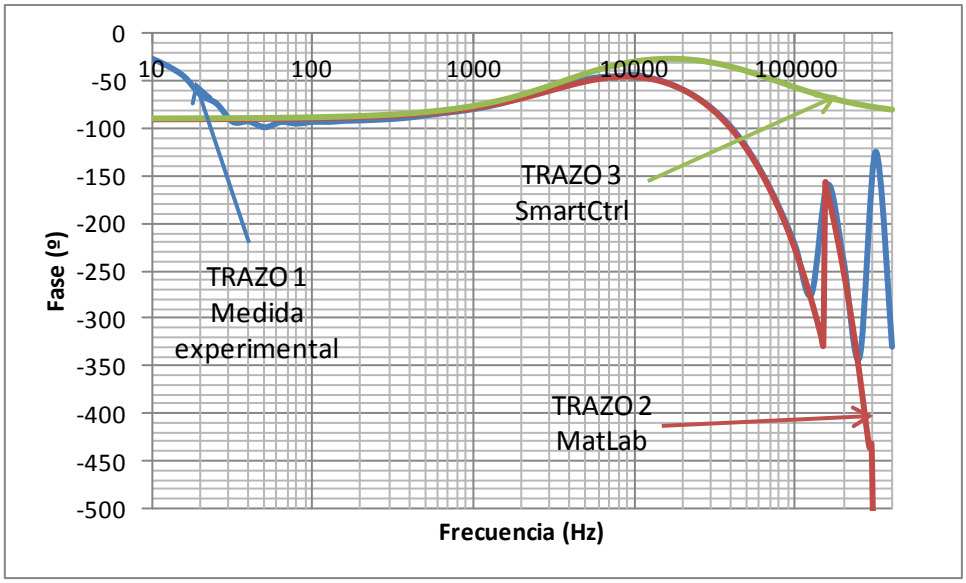


Figura 8-51 Figura 8-52 Resultados de fase del regulador tipo2_3

Los resultados obtenidos para ganancia y fase son muy buenos, excepto para frecuencias bajas, entre 10 y 60Hz, donde la medida sale mal por los problemas en el sistema de medida descritos en el apartado 8.1. A partir de ahí y hasta los 400kHz del barrido, los resultados son muy satisfactorios. En alta frecuencia se aprecia el efecto de ser un control digital, dado que la medida sigue perfectamente el trazo del cálculo con Matlab.

8.3.2.3.3 Lazo tipo2_3

Por último los resultados para el lazo son:

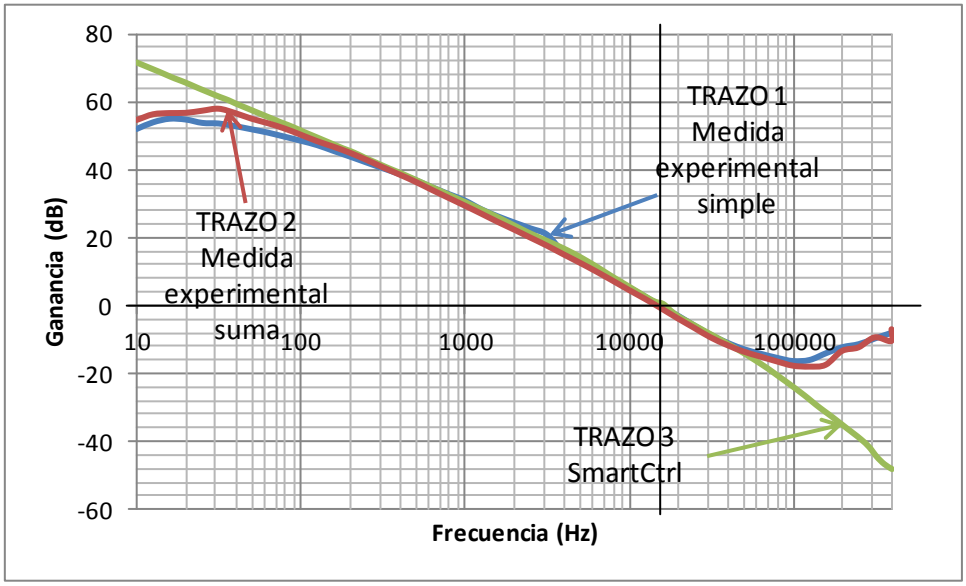


Figura 8-53 Resultados de ganancia del lazo abierto tipo2_3

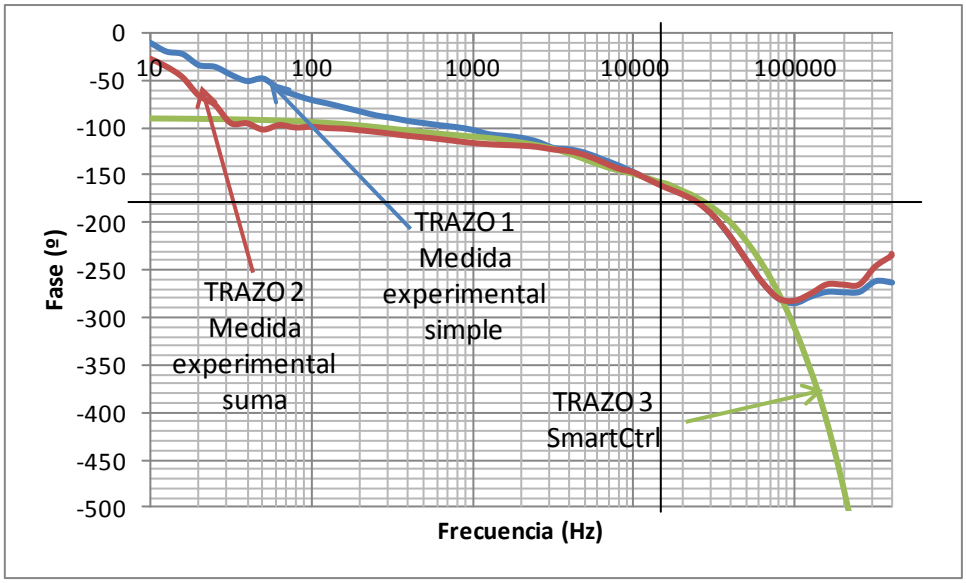


Figura 8-54 Resultados de fase del lazo abierto tipo2_3

Las medidas experimentales simples dan:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	15,74	15,85	16,6511
MF (º)	180-162,1 = 17,9	180-162,8 = 17,2	19,62

Tabla 8-8 Resultados del tipo2_3

Los resultados obtenidos para este tercer regulador son de nuevo satisfactorios. La ganancia experimental simple se ajusta bien a lo teórico para un amplio rango de frecuencias. La fase de esta medida, no alcanza los valores teóricos hasta cerca de 1kHz, sin embargo a partir de ahí su comportamiento es el esperado hasta que aparece como en todos los casos el efecto del sensor. Las medidas suma dan unos resultados muy positivos y siguen bien lo esperado tanto para la ganancia como para la fase.

8.3.2.4 Conclusiones reguladores Tipo2

Como en el caso de los reguladores PI, el proceso de diseño e implementación de reguladores en el dsPIC, ha dado unos resultados experimentales muy satisfactorios, con lo que se considera validada la metodología seguida para el diseño de los mismos.

De nuevo se han visto sobre las medidas los mismos problemas para bajas y altas frecuencias. Para las bajas, en medidas con grandes ganancias, es decir mucha diferencia entre lo medido entre Referencia y Test del analizador, los resultados no siguen correctamente lo teórico para el regulador y el lazo. Y en altas frecuencias aparece el fenómeno provocado por el comportamiento de alta frecuencia del sensor (divisor resistivo), que no es un error de la medida sino un fenómeno real integrado en el lazo. Para todo el resto de frecuencias el resultado obtenido ha sido muy positivo.

De nuevo se insiste en que es posible diseñar controles de tipo2 con ayuda de SmartCtrl para luego implementarlos en un montaje comercial real basado en dsPIC, y obtener

unos resultados similares a lo que teóricamente se diseña con SmartCtrl, al margen de ciertas discrepancias que pueden aparecer en el método de medida.

8.3.3 Reguladores tipo 3

Para terminar se mostraran las medidas experimentales obtenidas para los tres reguladores de tipo3, cuyas FDT tienen la forma:

$$G(s) = \frac{Y(s)}{X(s)} = \frac{b_2 s^2 + b_1 s + 1}{a_3 s^3 + a_2 s^2 + a_1 s} \qquad \text{Ec. 8-10}$$

$$G(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}} \qquad \textit{Ec. 8-11}$$

Los márgenes de fase y frecuencias de cruce seleccionados para validar los diseños de tipo3 son lo que se muestran en la Figura 8-55.

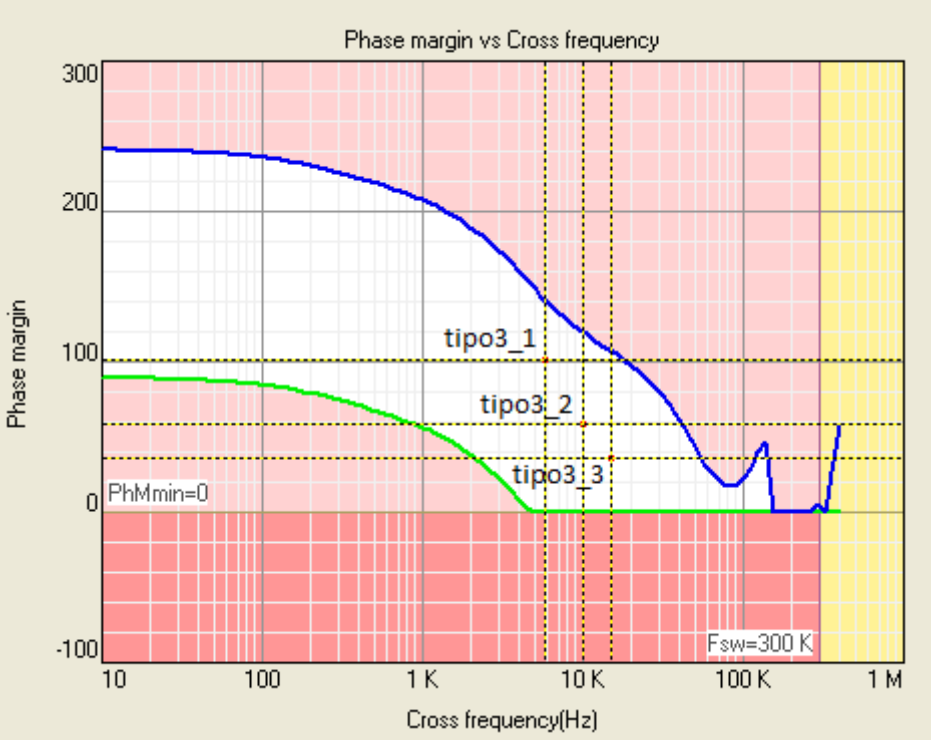


Figura 8-55 Mapa de soluciones para controles tipo 3

Para el caso de los tipo3, estos son los tres reguladores que se han medido. El área del mapa de soluciones que cubren estos tres puntos es menor que en los casos anteriores. Esto es así, porque aunque implementar reguladores más separados fue posible, a la hora de medir las respuestas frecuenciales de los mismos, los resultados diferían notablemente de lo esperado. Estos reguladores funcionaban en el dominio del tiempo y corregían la salida ante variaciones de V_{in}, sin embargo las medidas con el analizador Venable 3235 no fueron posibles para reguladores alejados de esta zona.

En concreto para el caso del tipo3_3 las medidas se tuvieron que realizar con el analizador HP 4194A en lugar del Venable 3235, ya que el segundo daba resultados incoherentes y sin embargo con el HP 4194A salían razonablemente bien, como se esperaba.

Por lo tanto para terminar se comentará que las medidas para los reguladores de tipo3 han resultado especialmente difíciles de tomar, llegando incluso en uno de los casos a tener que tomar medidas con el analizador HP 4194A, que ha demostrado ser más lento pero más fiable que el Venable para estos lazos con control de tipo3.

Para continuar con la exposición de los resultados experimentales, los márgenes de fase y frecuencias de cruce seleccionadas han sido:

	tipo 3_1	tipo 3_2	tipo 3_3
Fc (kHz)	5,7708	10	15,1364
MF (º)	101,448	58,896	36

Tabla 8-9 Frecuencia de cruce (fc) y margen de fase (MF) para lazos de tipo 3

En el caso de los lazos de tipo3, como se introdujo en el apartado 5.1.3, en la planta importada está incluido el efecto del sensor para altas frecuencias, ya que para bajas y medias se hizo un ajuste que lo anulaba. Por lo tanto para introducir este efecto intrínseco del sensor que está en todos los lazos probados, se debe introducir en el bloque sensor del SmartCtrl la siguiente ganancia.

$$G_{dsPIC} \cdot G_{sensor} = \frac{1}{2} \times 32 \times \frac{1023}{3{,}3} \times \frac{1}{3205} \times 0{,}6309 = 0{,}976469 \qquad \textit{Ec. 8-12}$$

En ella se introducen las ganancias descritas en el apartado 7.1.2.2.6 y además 0,6309, que ajusta precisamente la ganancia que fue introducida en el ajuste de la planta para en estos casos del tipo3, para tener contemplado el efecto del sensor a alta frecuencia (20xlog(0.6309)=-4dB). Por lo tanto para los tres ejemplos de tipo3, la ganancia que se fija en el apartado del sensor de SmartCtrl es, 0,976469.

8.3.3.1 Tipo3_1

El primer caso de los lazos con reguladores de tipo3 tiene las siguientes características de frecuencia de cruce y margen de fase:

Frecuencia de cruce (Fc): 5,7708kHz
Margen de Fase (MF): 101.448º

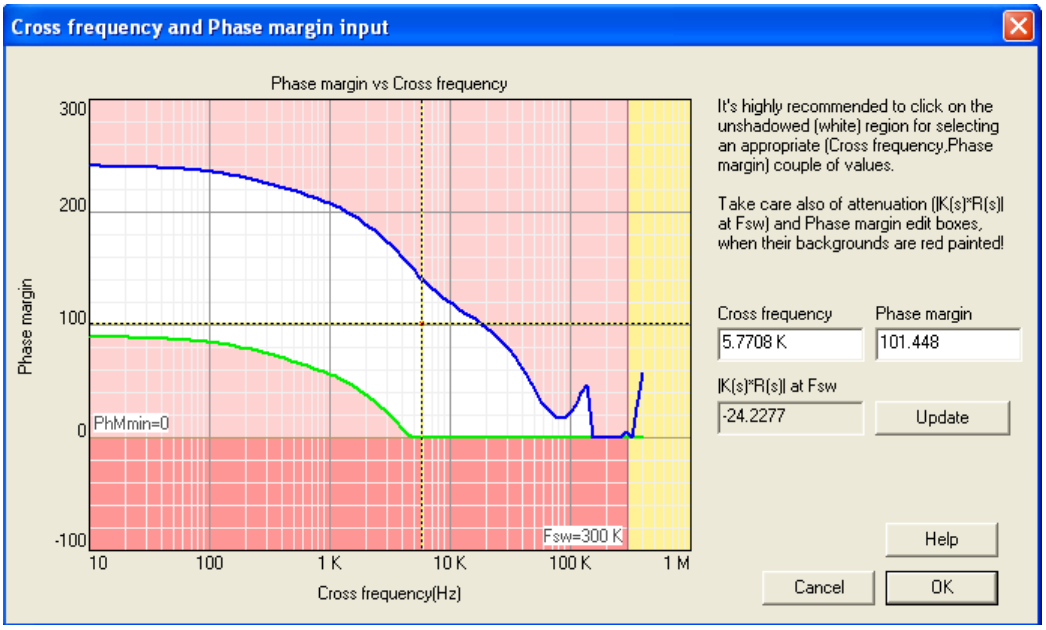


Figura 8-56 Mapa de soluciones tipo3_1

8.3.3.1.1 Planta tipo3_1

Los resultados de la medida experimental de la planta han sido:

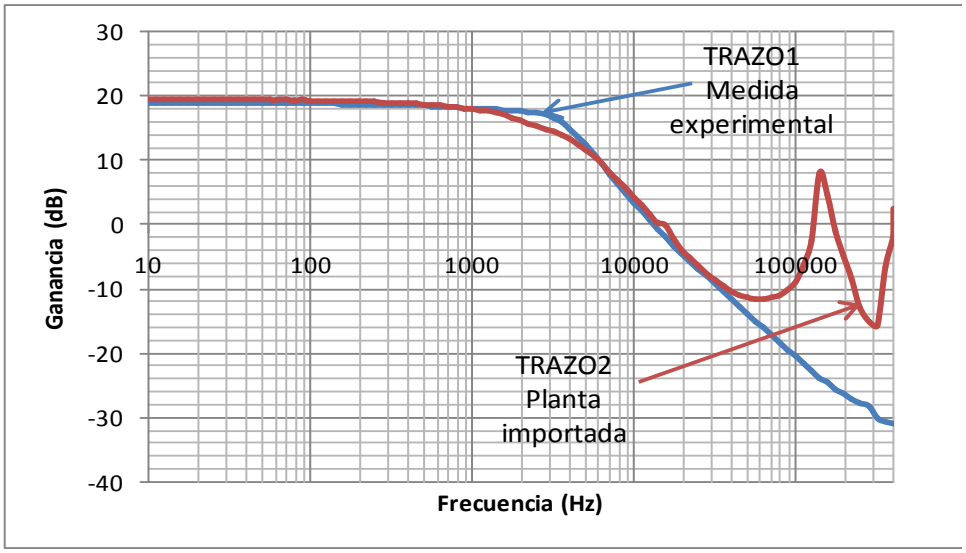


Figura 8-57 Resultados de ganancia de la planta tipo3_1

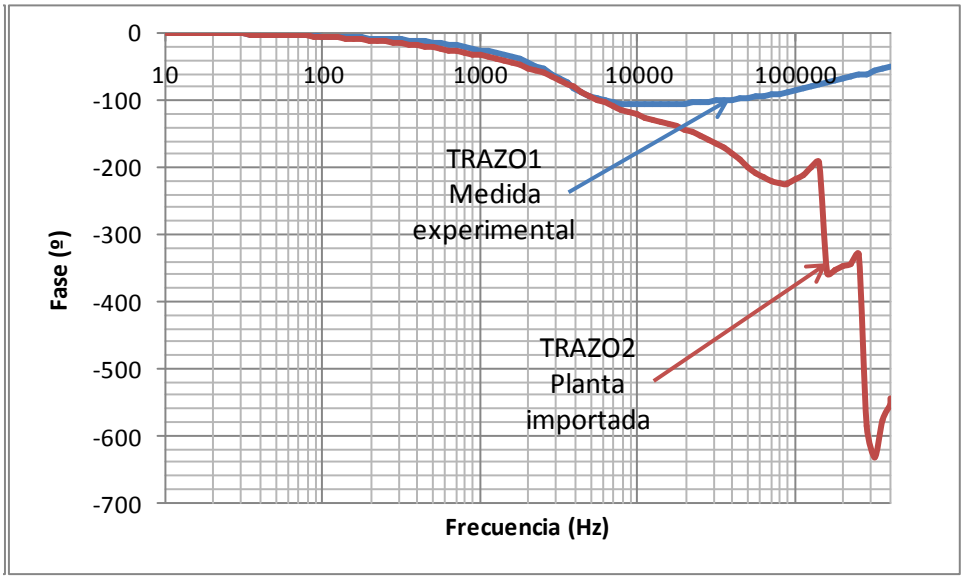


Figura 8-58 Resultados de la fase de la planta tipo3_1

Como se ve en la figura hasta 30kHz en ganancia y 8kHz en fase los resultados experimentales y los importados son razonablemente parecidos. Sin embargo a partir de 8kHz aproximadamente para la fase, el efecto del retardo se empieza a hacer muy significativo y se aleja de la medida experimental, además de poderse apreciar el efecto sensor a alta frecuencia. En la ganancia se puede ver mejor el efecto de la alta frecuencia introducida por el sensor. En este caso el retardo no existe y se ve como a partir de 30kHz, el efecto del sensor aleja lo medido de lo importado a SmartCtrl que incluye el sensor. En la medida experimental, no aparece este efecto porque pertenece al sensor y no a la planta, pero como se explica en el apartado 5.1.3, pero al importar a SmartCtrl si se incluye en la planta este efecto.

En los ejemplos anteriores mostrados de PI y tipo2, esta diferencia en la planta para la ganancia y fase no aparece ya que los anteriores no incluían en la planta el efecto del sensor. En esos casos, el efecto del sensor se introduce mediante el bloque sensor de SmartCtrl, que no incluye el efecto de alta frecuencia que sí será mostrado en estos casos de tipo3.

8.3.3.1.2 Regulador tipo3_1

Las medidas para el primer regulador de tipo3 son:

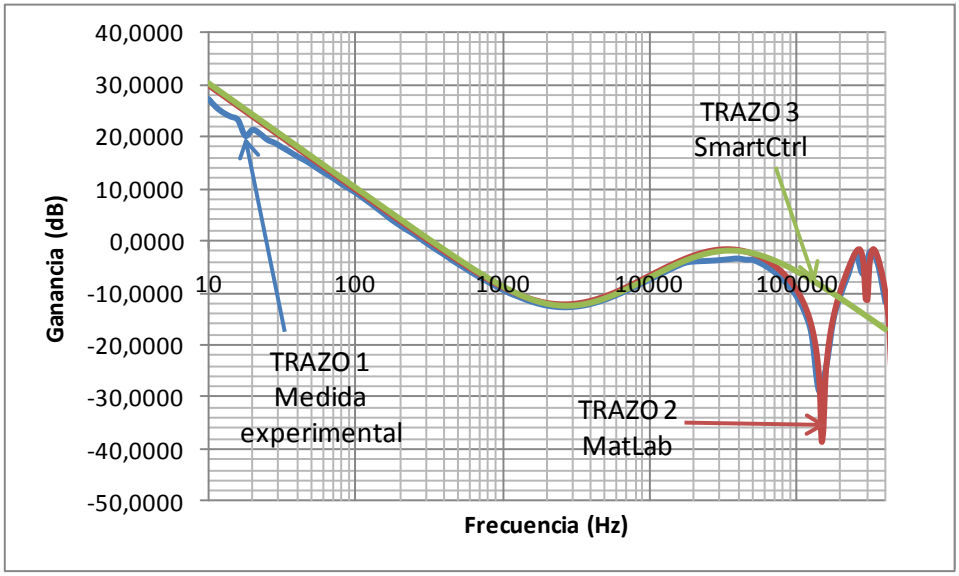


Figura 8-59 Resultados de ganancia del regulador tipo3_1

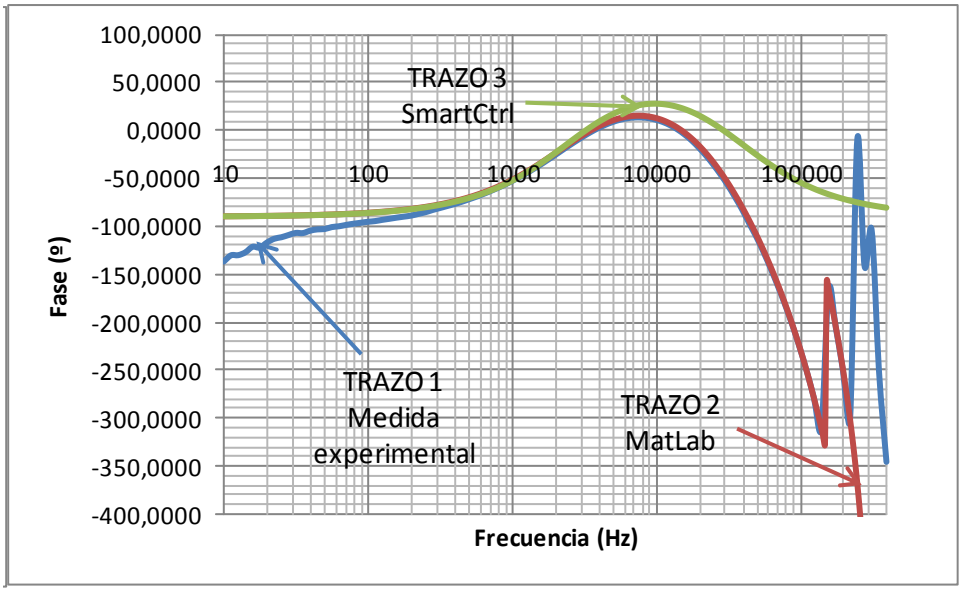


Figura 8-60 Resultados de fase del regulador tipo3_1

Los resultados obtenidos de las medidas siguen con buena precisión los valores teóricos que marcan MatLab y SmartCtrl. Como siempre, a bajas frecuencias la medida no es del todo buena y es a partir de 60Hz para ganancia y 100Hz para la fase, que lo medido comienza a seguir lo teórico. En cualquier caso para un amplio margen de frecuencias el comportamiento del controlador implementado es muy bueno.

Se puede ver levemente que entre 20kHz y 50kHz aproximadamente la ganancia se despega un poco de lo teórico. Este fenómeno se produce si la amplitud de la perturbación es demasiado grande. Si al hacer el barrido esa perturbación hubiese tenido menos amplitud, ese defecto no se habría visto reflejado en la medida. Por el contrario, si se hubiese bajado la

amplitud, en otras zonas la medida habría sido peor. De ahí la importancia de fijar la amplitud de la perturbación para tener un buen compromiso en los resultados para el mayor rango de frecuencias del barrido. Este fenómeno se ha dado habitualmente en el máximo local de ganancia a alta frecuencia de estos reguladores de tipo3.

8.3.3.1.3 Lazo tipo3_1

Los resultados para el lazo son:

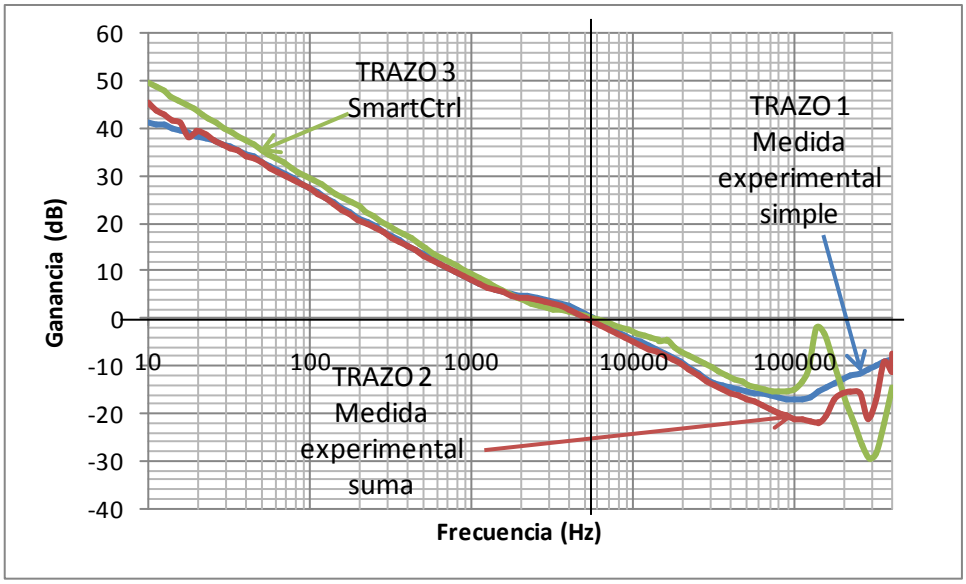


Figura 8-61 Resultados de ganancia del lazo abierto tipo3_1

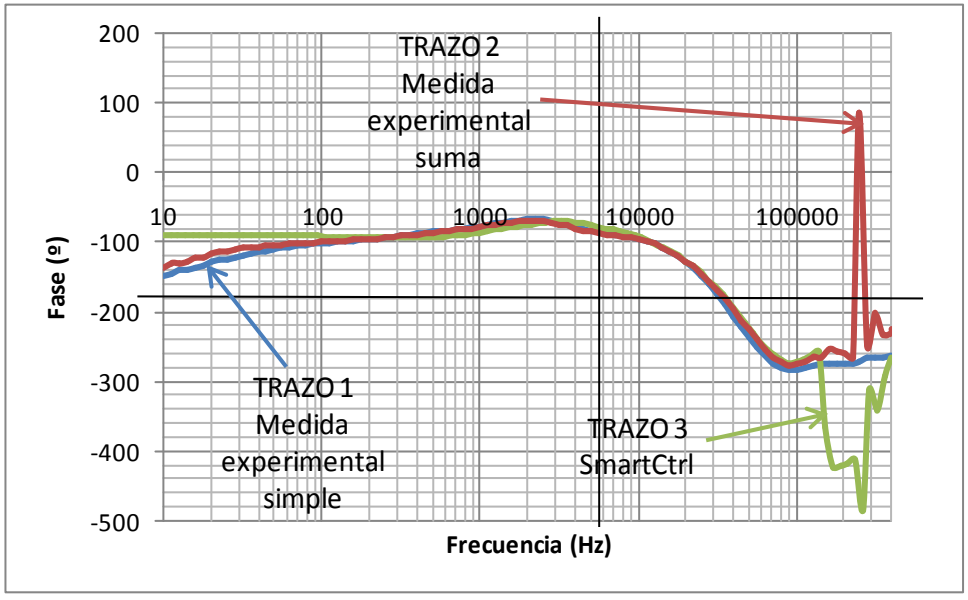


Figura 8-62 Resultados de fase del lazo abierto tipo3_1

Las medidas experimentales simples dan:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	5,6	5,25	5,7708
MF (º)	180-85,48 = 94,52	180-87,49 =92,51	101,448

Tabla 8-10 Resultados del tipo3_1

En este primer ejemplo de tipo3 los resultados son también favorables para la mayor parte del rango de frecuencias medidas. A diferencia de los casos anteriores, donde a alta frecuencia la ganancia teórica de SmartCtrl se alejaba rápidamente de las medidas, ahora con el efecto del sensor a alta frecuencia introducido en la planta importada, la medida teórica sigue aproximadamente a lo medido. Es cierto que con relevantes diferencias, pero ya no se aleja progresivamente de lo medido desligándose completamente lo medido de lo teórico. Por consiguiente, este método de introducción del sensor en el lazo teórico de SmartCtrl, es razonable. Para la fase a alta frecuencia es difícil apreciar nada ya que al estar tan cerca de la frecuencia de conmutación y muestreo, la fase es difícil de medir y pueden aparecer efectos de muestreo como el aliasing que pueden desvirtuar la medida.

8.3.3.2 Tipo3_2

El siguiente ejemplo de tipo3 tiene las siguientes características de Fc y MF, las cuales son bastante razonables para el convertidor que se tiene y la frecuencia de conmutación del circuito de control.

Frecuencia de cruce (Fc): 10,00kHz
Margen de Fase (MF): 58,896º

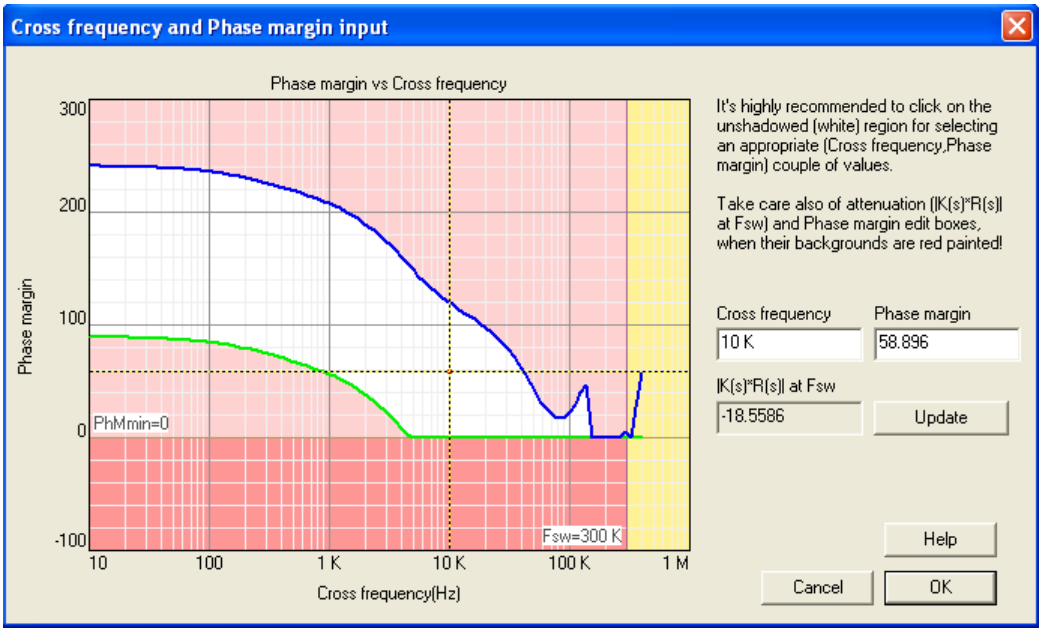


Figura 8-63 Mapa de soluciones tipo3_2

8.3.3.2.1 Planta tipo3_2

Los resultados de la planta han sido los siguientes:

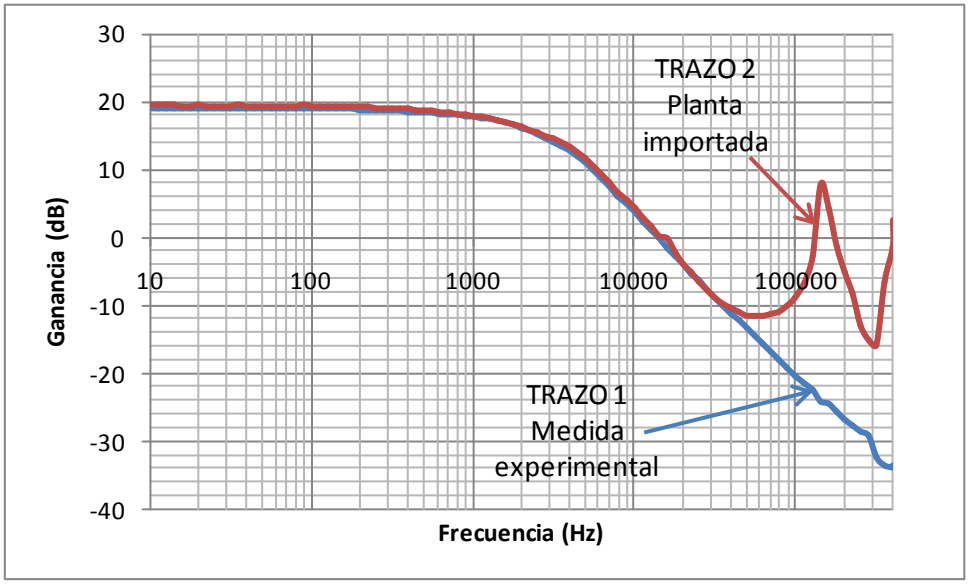


Figura 8-64 Resultados de ganancia de la planta tipo3_2

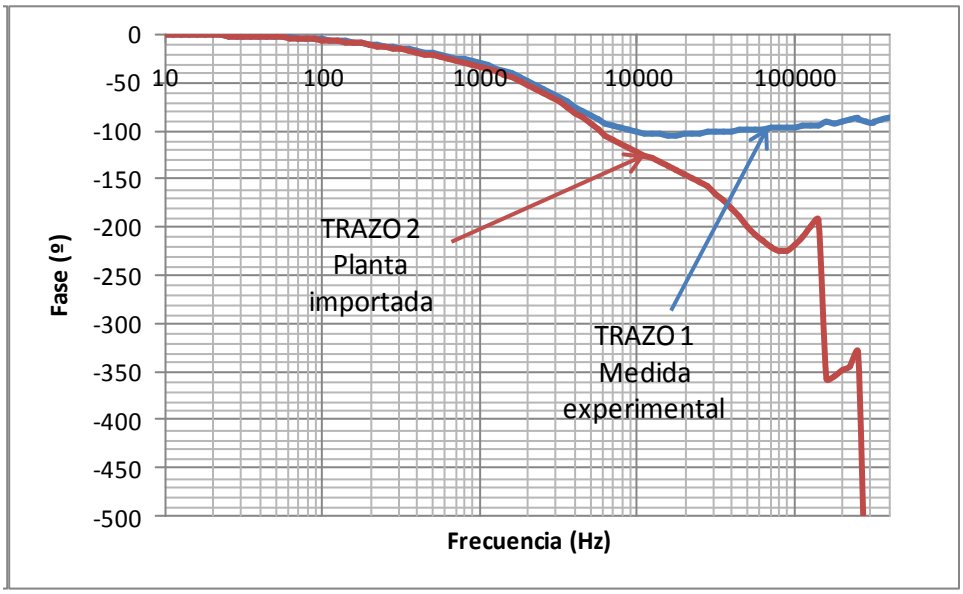


Figura 8-65 Resultados de fase de la planta tipo3_2

Como en el caso anterior, la planta importada incluye el efecto del sensor, que hace que el comportamiento de la planta a alta frecuencia sea muy distinto de lo esperado y de lo medido de la planta real.

Este efecto se aprecia mejor en la ganancia, ya que la fase está de por sí afectada por el retardo digital del lazo que es introducido en SmartCtrl en la planta importada.

8.3.3.2.2 Regulador tipo3_2

Los resultados experimentales del regulador tipo3_2 han sido los siguientes:

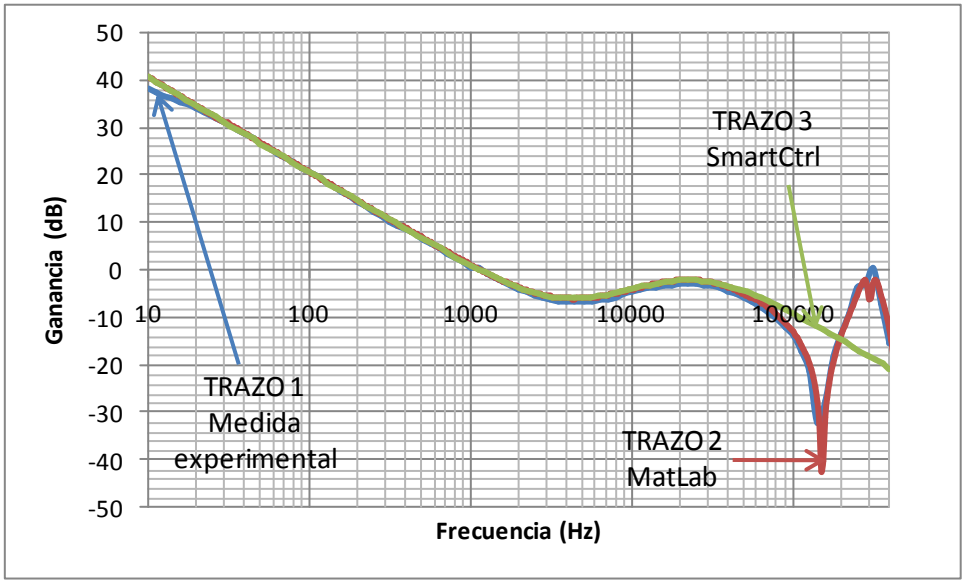


Figura 8-66 Resultados de ganancia del regulador tipo3_2

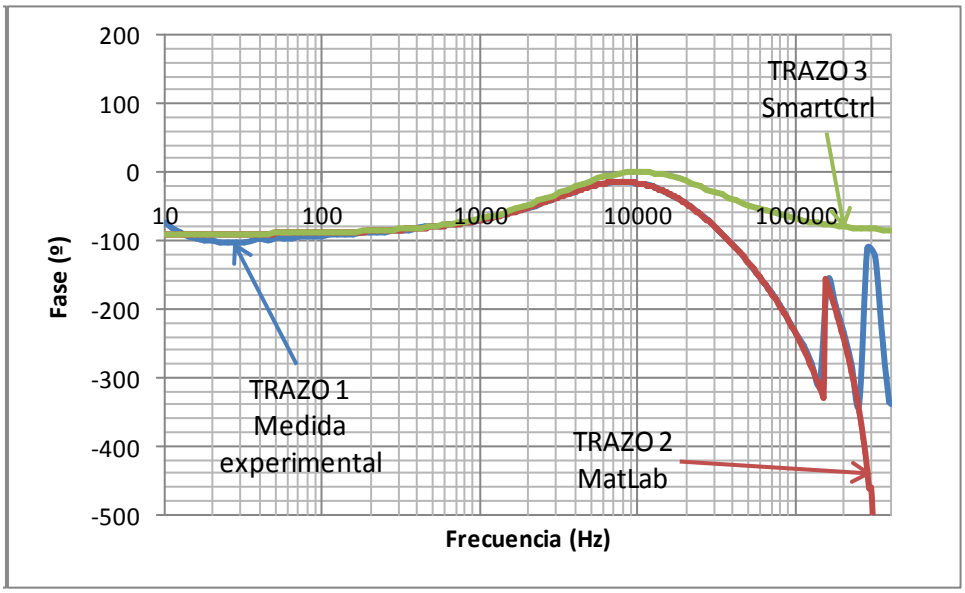


Figura 8-67 Resultados de fase del regulador tipo3_2

Los resultados obtenidos para este caso son muy buenos. En ganancia, lo medido sigue perfectamente lo teórico que nos da MatLab, que a su vez sigue lo diseñado en continuo mediante SmartCtrl. Y para la fase de nuevo los resultados son muy satisfactorios, sólo se aprecia una pequeña discrepancia entre lo medido y lo teórico para frecuencias bajas por debajo de 50 Hz. En el resto de frecuencias, la medida sigue perfectamente el regulador discreto teórico de MatLab.

8.3.3.2.3 Lazo tipo3_2

Los resultados finales de este lazo de control tipo3 son los siguientes:

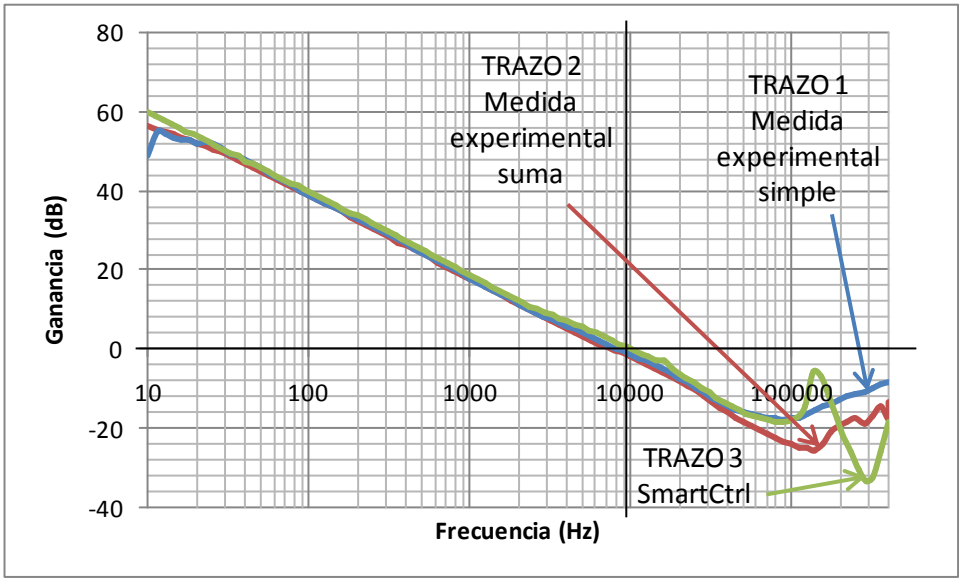


Figura 8-68 Resultados de ganancia del lazo abierto tipo3_2

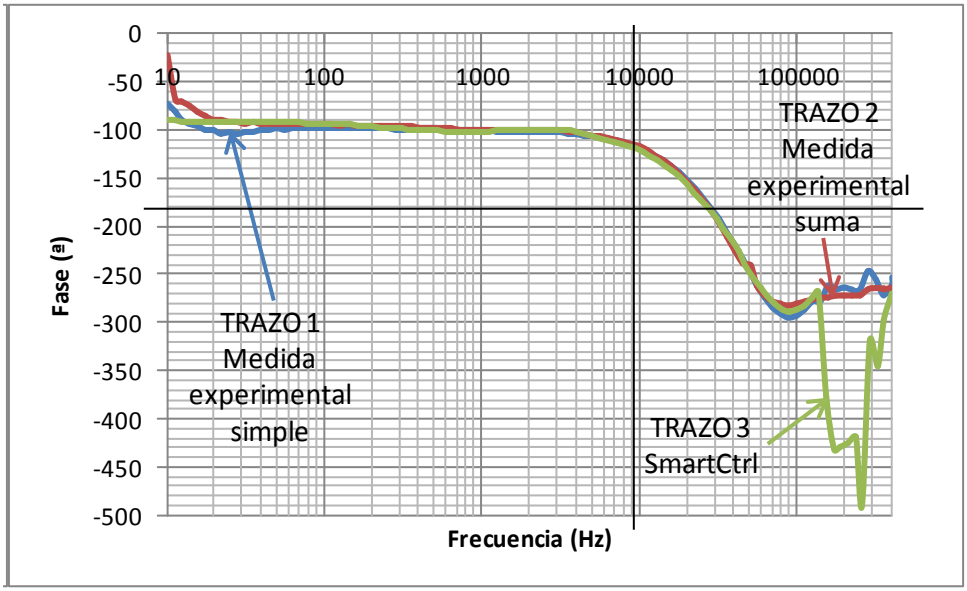


Figura 8-69 Resultados de fase del lazo abierto tipo3_2

Las medidas experimentales simples dan:

	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	8,9	8,1	10
MF (º)	180-114,6 = 65,4	180-112,9 = 67,1	58,896

Tabla 8-11 Resultados del tipo3_2

Los valores de Fc y MF teóricos y experimentales quedan un poco alejados ya que en escala logarítmica, desde 8kHz hasta 10kHz, hay poca diferencia por lo que aunque las gráficas pasen muy juntas en los tres casos, los valores concretos quedan un poco alejados. A la hora de cruzar con 0 dB, las medidas experimentales lo hacen ligeramente antes que el trazo teórico, y esa ligera diferencia en los trazos a esas frecuencias supone cruzar a 8 ó 9kHz en lugar de los 10kHz esperados teóricamente. Con la fase se tiene el mismo efecto. Sin embargo en general las medidas experimentales se ajustan mucho al trazo teórico que indica SmartCtrl, por lo que este ejemplo justifica que el proceso de diseño empleado para el regulador es válido, como en todos los casos vistos hasta ahora.

8.3.3.3 Tipo3_3

Para este ejemplo las medidas experimentales fueron tomadas con el analizador HP. Debido a que este analizador es muy lento para frecuencias bajas, los barridos se han iniciado desde 100Hz. Precisamente a estas bajas frecuencias por debajo de 100Hz es donde se han visto discrepancias entre lo medido y lo teórico en casos anteriores, por lo cual los resultados en este caso serán previsiblemente buenos ya que se parte desde los 100Hz hasta los 400kHz.

En este último caso de tipo3, las características de Fc y MF son:

Frecuencia de cruce (Fc):	15,1364kHz
Margen de Fase (MF):	36º

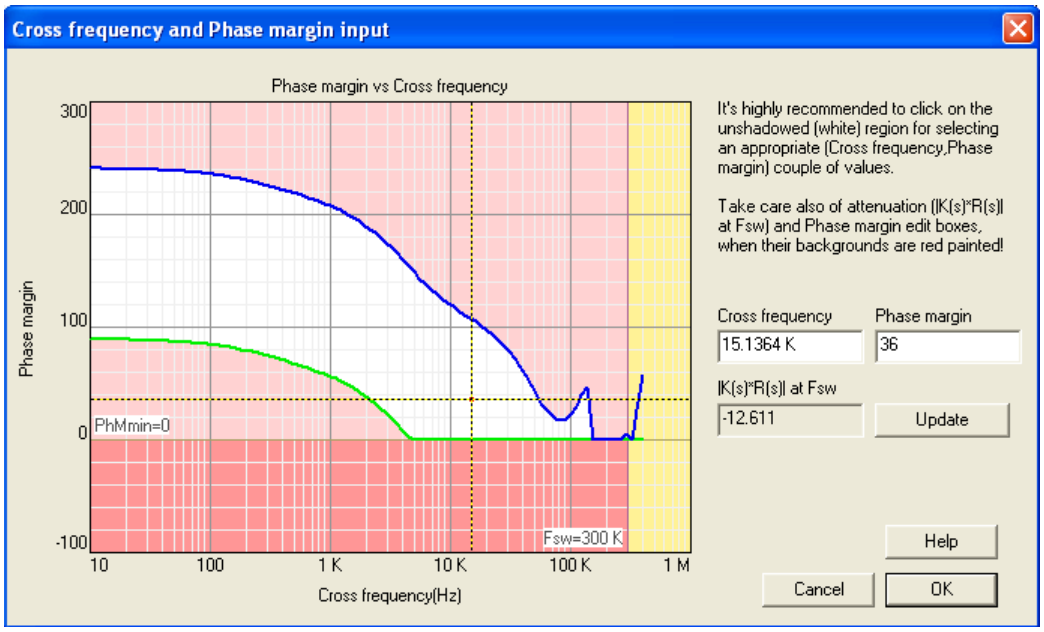


Figura 8-70 Mapa de soluciones tipo3_3

8.3.3.3.1 Planta tipo3_3

Para la planta se han obtenido los siguientes resultados:

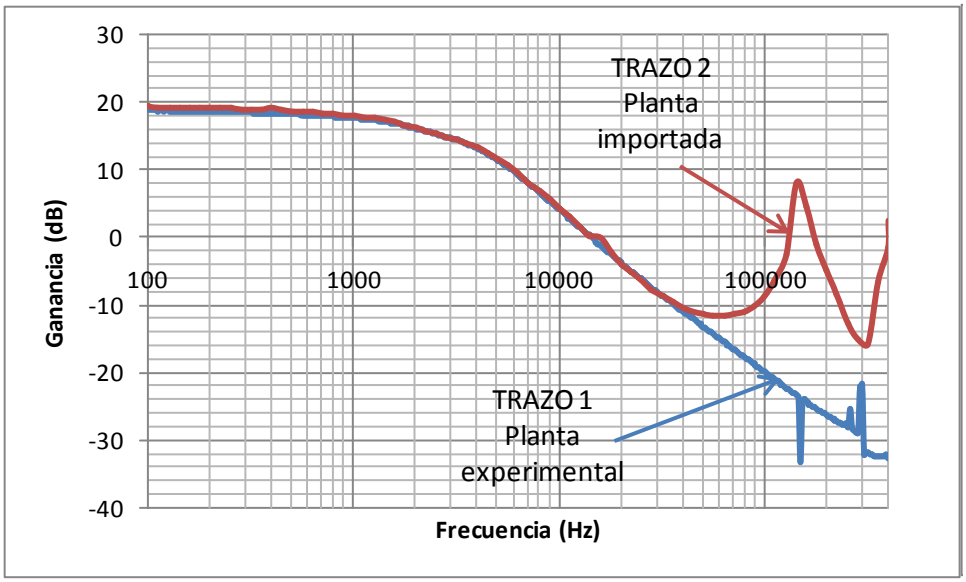


Figura 8-71 Resultados de ganancia de la planta tipo3_3

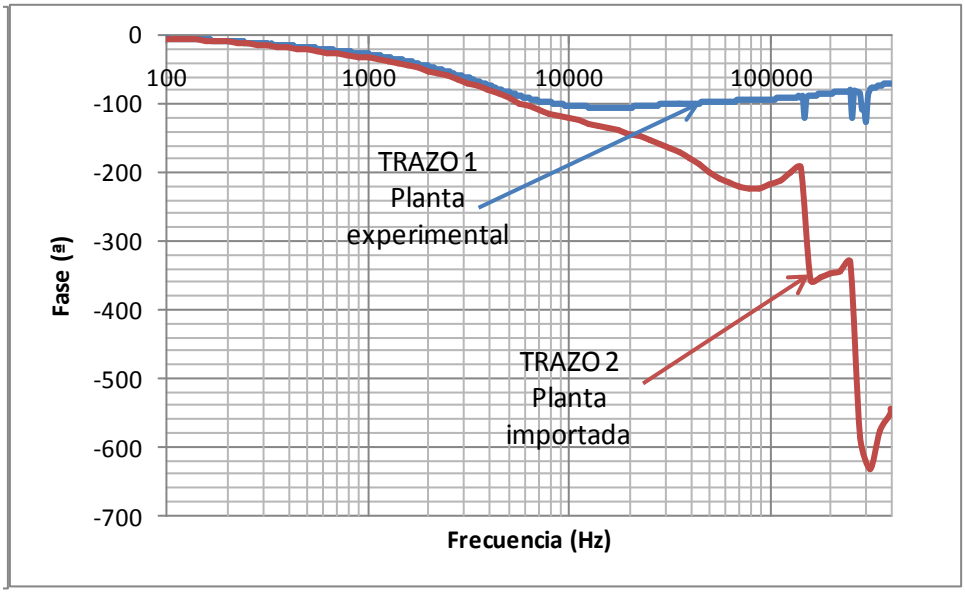


Figura 8-72 Resultados de la fase de la planta tipo3_3

Como en los demás casos tipo3, la planta importada incluye el efecto del sensor. Por este motivo a partir de 40kHz, la ganancia medida y la importada difieren. Para la fase ocurre lo mismo que en los casos anteriores, al retardo se añade el efecto del sensor.

8.3.3.3.2 Regulador tipo3_3

Para este último caso tipo3, los resultados obtenidos han sido:

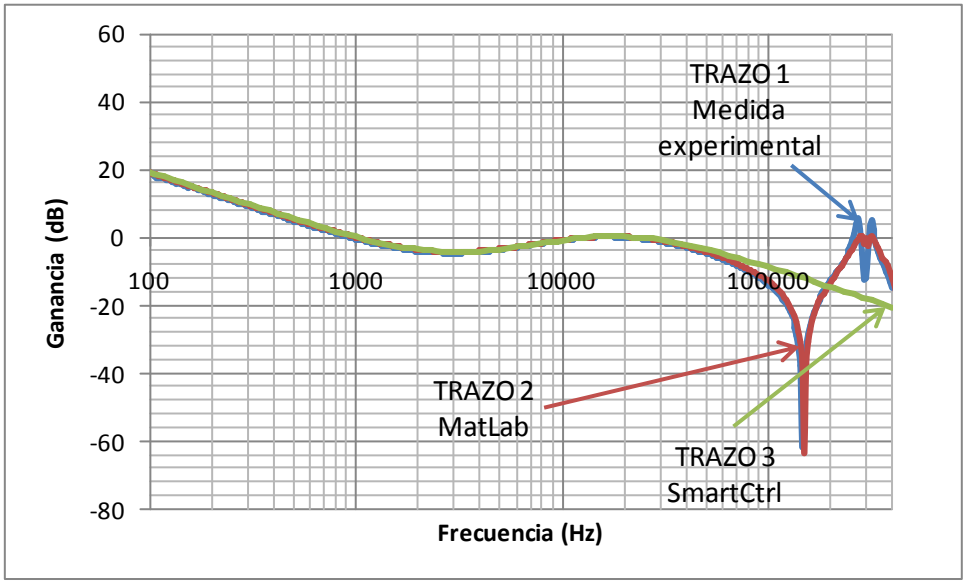


Figura 8-73 Resultados de ganancia del regulador tipo3_3

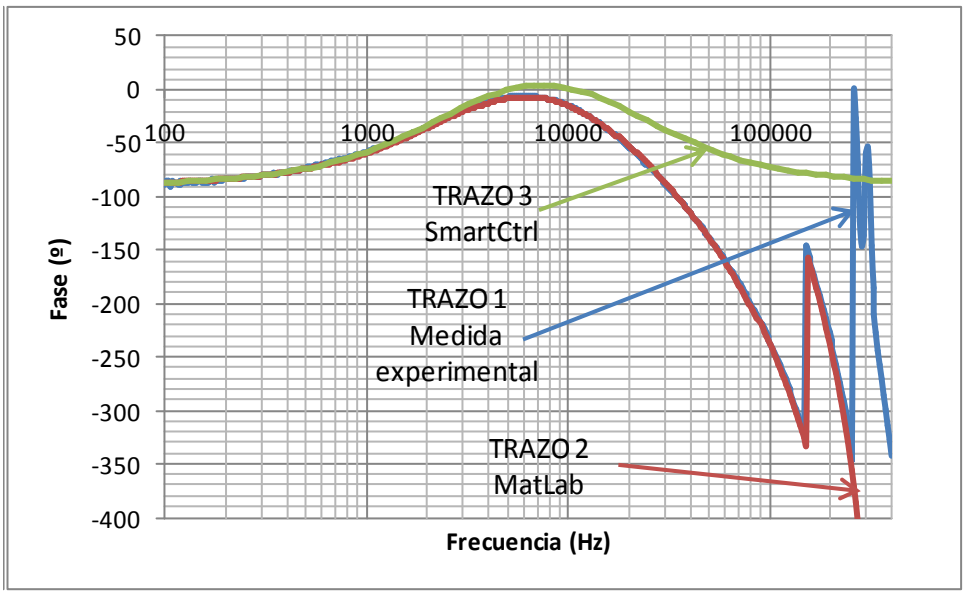


Figura 8-74 Resultados de fase del regulador tipo3_3

Los resultados obtenidos desde 100Hz hasta 400kHz son muy buenos. La medida sigue perfectamente lo teórico indicado por MatLab en discreto y lo continuo del SmartCtrl. Para la fase igual, los resultados son muy buenos. Es posible que para frecuencias menores a 100Hz hubiera discrepancias, pero al haber usado en estas medidas el analizador HP, estas bajas frecuencias no han sido tomadas.

8.3.3.3.3 Lazo tipo3_3

Los resultados finales de este lazo de control tipo3 son los siguientes:

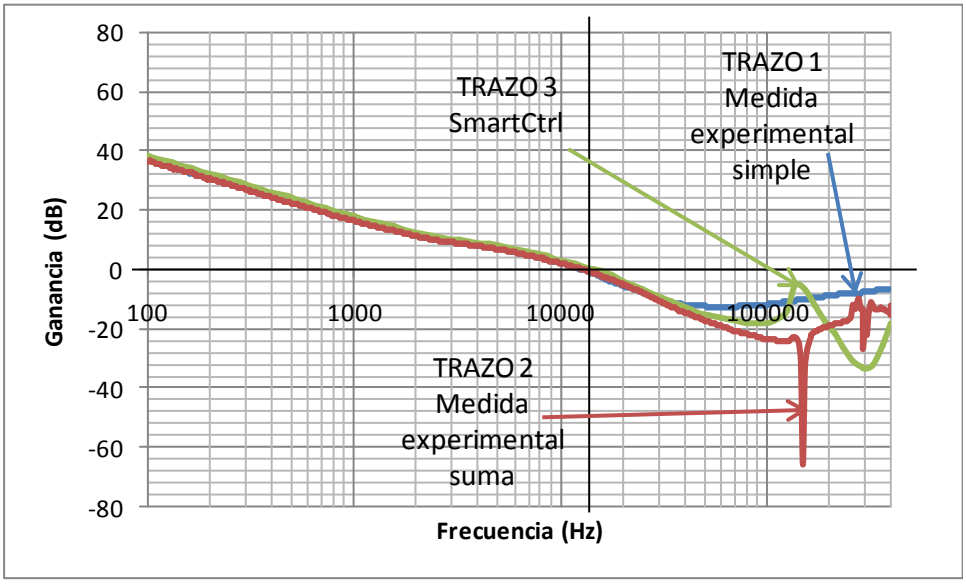


Figura 8-75 Resultados de ganancia del lazo abierto tipo3_3

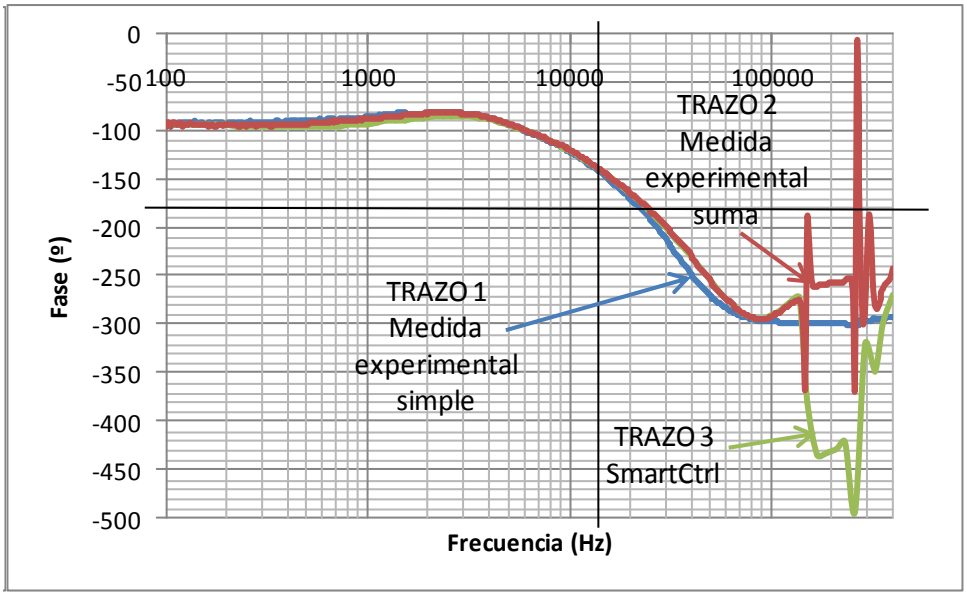


Figura 8-76 Resultados de fase del lazo abierto tipo3_3

Las medidas experimentales simples dan los siguientes valores:

9	Medidas simples	Medidas suma	SmartCtrl
Fc (kHz)	12,79	12,98	15,1364
MF (º)	180-135,3 = 44,7	180-134,4 = 45,6	36

Tabla 8-12 Resultados del tipo3_3

Los valores obtenidos de ganancia y fase experimentales son muy parecidos a los teóricos, aunque si hay una pequeña discrepancia para los valores de Fc y MF. Sin embargo en general los resultados experimentales son similares a lo esperado por lo que de nuevo en este último caso, queda validado el proceso de diseño de reguladores digitales basados en dsPIC mediante SmartCtrl.

9.1.1.1 Conclusiones reguladores Tipo3

Como en los casos anteriores de tipo2 y PI, los resultados experimentales han confirmado que la metodología de diseño de reguladores digitales descrita en este proyecto, es válida para un amplio rango de lazos. Para ello mediante tres ejemplos con distintas características de Fc y MF, se ha demostrado que su funcionamiento es razonablemente similar a lo teórico diseñado con ayuda de la herramienta SmartCtrl para el diseño de los reguladores.

En el caso concreto de los tipo3, se ha intentado introducir el fenómeno de alta frecuencia del sensor. Como resultado de ello se ha conseguido que el comportamiento a altas frecuencias reflejado en SmartCtrl, se parezca más a lo experimental que en caso de no añadirlo. Sin embargo con este cambio no se consigue que lo experimental y lo teórico sean iguales para altas frecuencias (mayores de 100kHz), lo que sí ocurre para bajas y medias (desde 100Hz a 100kHz). Esto es así debido principalmente a que las altas frecuencias son más difíciles de medir (referencia respecto a test) y además pueden aparecer fenómenos de muestreo al trabajar con lazos digitales.

Finalmente se ha conseguido con estos tres ejemplos de reguladores tipo3, validar el proceso de diseño basado en SmartCtrl para diseñar lazos digitales e implementarlos en un montaje con convertidores reales y controlador basado en dsPIC.

9.1.1.2 Resultados finales y conclusiones de los resultados experimentales

Nombre del regulador	Tipo	SmartCtrl		Lazo medido		Suma de componentes del lazo	
		Fc	MF	Fc	MF	Fc	MF
PI_1	PI	1,15 kHz	84º	1,26 kHz	95,18º	1,12kHz	86,58º
PI_2	PI	4,22 kHz	53º	4,46 kHz	40,4º	4,2 kHz	47º
PI_3	PI	10,23 kHz	39,23º	9,5 kHz	32,2º	9,5 kHz	35,3º
Tipo2_1	Tipo2	1,49 kHz	98,17º	2,8 kHz	72,4º	1,41 kHz	101,2º
Tipo2_2	Tipo2	4,09 kHz	49º	4,05 kHz	45,2º	3,65 kHz	54,1º
Tipo2_3	Tipo2	16,65 kHz	19,62º	15,74 kHz	17,9º	15,85 kHz	17,2º
Tipo3_1	Tipo3	5,77 kHz	101,45º	5,6 kHz	94,52º	5,25 kHz	92,51º
Tipo3_2	Tipo3	10 kHz	58,9º	8,9 kHz	65,4º	8,1 kHz	67,1º
Tipo3_3	Tipo3	15,13 kHz	36º	12,79 kHz	44,7º	12,98 kHz	45,6º

Tabla 8-13 Resumen de los resultados experimentales

Tras la obtención de los resultados experimentales de estos nueve reguladores distintos implementados en dsPIC, se obtiene como conclusión, que es posible diseñar e implementar rápidamente controles en lazo de cerrado para convertidores de potencia conmutados basados en dsPIC, con la ayuda de la herramienta CAD de SmartCtrl. Estos nueve ejemplos de controles han sido diseñados siguiendo el proceso que se describe en la Tabla 5-1 y que basa el diseño de reguladores en el uso de la herramienta SmartCtrl. Una vez implementados los controles, se han medido dando los resultados descritos en este apartado.

Estos resultados experimentales han demostrado que para un amplio rango de frecuencias, desde 100Hz hasta 30kHz, los resultados teóricos obtenidos de la herramienta de diseño SmartCtrl y lo finalmente implementado y medido en el montaje experimental, dan resultados muy similares. Con estos resultados, se puede afirmar que en torno a estas frecuencias se pueden diseñar lazos de control con SmartCtrl, implementarlos en un control digital basado en dsPIC y obtener unos resultados experimentales con un grado de similitud entre lo teórico y lo real, muy alto.

En general, las medidas para bajas frecuencias de los componentes del lazo con polo en el origen (10-100Hz), dan resultados peores que para frecuencias medias. Esto es debido a que el sistema de medida tiene dificultades para hacer las medidas cuando se tienen ganancias muy altas, como es el caso de estos bloques del esquema de control que tienen un polo en el origen. Por lo tanto el problema no es medir a frecuencias bajas, ya que la planta sale bien, sino hacer las medidas con ganancias muy altas.

Para frecuencias altas también hay diferencias entre lo medido y lo teórico de SmartCtrl y estas son principalmente debidas al montaje experimental. Este tiene un comportamiento a alta frecuencia difícil de modelar lo que hace difícil introducirlo en SmartCtrl. En las medidas experimentales esto se ha visto principalmente en los resultados de las medidas del sensor.

Por último hay que recordar que para poder comparar los resultados de SmartCtrl y los experimentales, hay que descontar a la medida las ganancias introducidas en el lazo por el uso de un regulador digital que realimenta la variable a controlar con un ADC y que modula la señal de control mediante una señal PWM. Por lo tanto a las medidas experimentales hay que descontarle estas ganancias descritas en el apartado 7.1.2.2 y que son propias del regulador digital dsPIC.

10 Conclusiones

Una vez realizados los trabajos descritos en esta memoria, se pueden obtener las siguientes conclusiones en base a los objetivos del proyecto descritos en el apartado 1.1 y los resultados experimentales mostrados en el capítulo 8

Mediante el proceso de diseño descrito en este proyecto y resumido en la Tabla 5-1, es posible diseñar e implementar reguladores digitales basados en dsPIC con ayuda de la herramienta CAD SmartCtrl, de una manera rápida y sencilla obteniéndose unos resultados experimentales muy próximos a los teóricos.

Como consecuencia de este objetivo principal, **se ha obtenido la validación experimental de los resultados que ofrece SmartCtrl.** Partiendo de los resultados de SmartCtrl y una vez adaptados para realizar un control digital, los resultados experimentales de respuesta en frecuencia obtenidos, muestran una similitud con los que indica SmartCtrl muy alta. Por lo tanto en el ámbito de este proyecto ha quedado demostrado que **basándose en los resultados de SmartCtrl, se puede diseñar un control para un convertidor conmutado real, de una manera rápida, sencilla y fiable**.

Los diseños realizados y documentados en este proyectos son además fácilmente reproducibles ya que se basan en hardware comercial (convertidores de potencia y tarjeta de control) y software fácilmente accesible (SmartCrtl, Matlab y MPLab)

10.1 Retardo introducido por el regulador

Como se ha explicado en la memoria, el retardo es introducido por el ADC y el PWM, ambos periféricos del dsPIC, que es quien hace la función de regulación. Sin embargo este retardo que afecta sensiblemente a la fase del lazo y con ello a la estabilidad del mismo, no se puede introducir directamente en la versión de SmartCtrl usada. Se ha tenido en cuenta añadiendo a la planta importada dicho retardo, aunque esté pertenece al regulador y no a la planta. Esta ha sido la manera de incluirlo en los cálculos de SmartCtrl y así poder fijar Frecuencia de Cruce y Margen de Fase a un lazo que incluye el retardo.

Por lo tanto de los trabajos realizados en este proyecto se propone incluir en la versión de SmartCtrl para controles digitales, la posibilidad de incluir un retardo en el regulador.

Este retardo debe incluir el periodo de cálculo de valores de control y el tiempo de actualización del nuevo valor de control en el PWM. Por ejemplo en el caso de este proyecto el retardo ha sido de 1,33xTPWM.

10.2 Introducción de las ganancias del control digital

Otro aspecto de especial importancia visto en el desarrollo de este proyecto, son las ganancias introducidas por los periféricos del dsPIC. El ADC, el PWM y los propios cálculos del dsPIC introducen ganancias en el lazo y estas deben ser tenidas en cuenta en SmartCtrl. En este proyecto, estas ganancias han sido introducidas a través del sensor de SmartCtrl, que además de incluir la propia ganancia del sensor (divisor de tensión), se ha usado para introducir el valor de estas ganancias del regulador digital.

Como siguiente conclusión, **se aconseja para la versión de SmartCtrl para control digital la posibilidad de incluir fácilmente en el lazo de control las ganancias que dependen de la resolución y fondo de escala del ADC, del módulo PWM y del propio DSP.**

10.3 Respuesta en frecuencia del sensor

Este último aspecto es interesante tanto para el SmartCtrl digital como el continuo según la experiencia de este proyecto. La versión de SmartCtrl usada en este proyecto incluye distintas opciones de sensor a introducir en el lazo. Sin embargo en el caso del sensor de este proyecto ninguna de ellas ha sido válida a altas frecuencias. Experimentalmente se ha medido la respuesta frecuencial del sensor y ha dado unos resultados en alta frecuencia imposibles de introducir en el lazo a través del sensor de SmartCtrl, siendo introducido este fenómeno a través de la planta importada como se hizo con el retardo (sólo en el las medidas experimentales de tipo3).

Por lo tanto, **para futuras versiones de SmartCtrl se aconseja incluir la posibilidad de introducir la respuesta en frecuencia del sensor** del mismo modo que la de la planta, además de las opciones ya incluidas en SmartCtrl.

10.4 Trabajos futuros

Aparte de los aspectos citados anteriormente, también son interesantes para profundizar en la línea de trabajo de este proyecto las siguientes cuestiones:

- Obtención de resultados experimentales para el convertidor elevador de la misma placa "*Buck*/*Boost*". Sobre este, se han probado distintos controles, pero no se han documentado ni detallado tan detalladamente como para el reductor.
- Estudio de los resultados experimentales frente a respuesta escalón, y comparación con el mismo de SmartCtrl. (La versión SmartCtrl usada, no mostraba la respuesta escalón con plantas importadas)
- Realización de controles digitales dsPIC para otros tipos de convertidores. Reductor-elevador, flyback, forward y otros reductores y elevadores prototipos
- Desarrollo de controles similares sobre plataformas de otros fabricantes, Texas Instruments, Freescale, etc.

11 Anexos

11.1 Ficheros de programas implementados en dsPIC

11.1.1Init .c

```
#include "p33FJ16GS502.h"

#include "Functions.h"

#include "dsp.h"

*/declaración de variable tipo tpid, para el reductor2*/

extern tPID Buck2VoltagePID;

/*reserva de memoria en zona x, donde se ubicaran los coeficientes de la fdt*/

fractional Buck2VoltageABC[7] __attribute__ ((section (".xbss, bss, xmemory")));

/*reserva de memoria en zona y, donde se ubicaran error y d actual y anteriores*/

fractional Buck2Voltage[7] __attribute__ ((section (".ybss, bss, ymemory")));

/*el tamaño depende del tipo de regulador a implementar, en este caso tipo3*/

/*coeficientes del denominador de la fdt, solo parte entera, con signo cambiado*/

#define PID_BUCK2_A3 Q15(+0.358)

#define PID_BUCK2_A2 Q15(-0.554)

#define PID_BUCK2_A1 Q15(+0.196)

/*coeficientes del numerador de la fdt, parte entera*/

#define PID_BUCK2_B0 Q15(+0.2649)

#define PID_BUCK2_B1 Q15(-0.2215)

#define PID_BUCK2_B2 Q15(-0.2631)

#define PID_BUCK2_B3 Q15(+0.2233)

/*funcion de transferencia implementada*/
```

```
//0.2649 z^3 - 0.2215 z^2 - 0.2631 z + 0.2233
//----------------------------------------
//   z^3 - 2.196 z^2 + 1.554 z - 0.3577
/*referencia1=3.3v, del sensor (divisor de tension)*/
/*5/(5+3.3)*3.3=1.988v--->(1.988*1024)/3.3v=617*/
/*ADCres=10, Q15res=15--->617*32=19744 = 0x4D20*/
#define PID_BUCK2_VOLTAGE_REFERENCE 0x4D20
/*referencia1=4.81v, del sensor (divisor de tension)*/
/*5/(5+3.3)*4.81=2.9v--->(2.9*1024)/3.3v=900*/
/*adcres=10, q15res=15--->900*32=28800 = 0x7080*/
#define PID_BUCK2_VOLTAGE_REFERENCE2 0x7080
#define BUCK2_SOFTSTART_INCREMENT
((PID_BUCK2_VOLTAGE_REFERENCE - PID_BUCK2_VOLTAGE_MIN) / 50 )
extern unsigned int TimerInterruptCount;
extern unsigned int flag;
extern unsigned int reference1;
extern unsigned int reference2;
void Buck2Drive(void)
{
 /*configuracion del módulo pwm del reductor2*/
IOCON2bits.PENH = 1;          /* pin pwm2h es cotrolado por el módulo pwm */
IOCON2bits.PENL = 1;          /* pin pwm2l es cotrolado por el módulo pwm */
IOCON2bits.PMOD = 0;
```

```
/* funcionamiento en modo complementario, reductor síncrono */

IOCON2bits.POLH = 0;            /* salida pwm2h, activa a nivel alto*/

IOCON2bits.POLL = 0;            /* salida pwm2h, activa a nivel alto */

IOCON2bits.OVRENH = 0

/* desactivada uso de este pin otras cosas que no sean la generacion del pwm */

 IOCON2bits.OVRENL = 0;

/* desactivada uso de este pin otras cosas que no sean la generacion del pwm */

IOCON2bits.OVRDAT = 0b00;       /* 00, solo valido si ovrenh = 1 o ovrenl = 1*/

PWMCON2bits.DTC= 0; /* dead time positivo, pwml y pwmh, no pueden coincidir */

 DTR2    = 0x30;                /* 30ns / 1.04ns = 28.85ns de dead time por un lado */

 ALTDTR2 = 0x30;

/* 30ns / 1.04ns = 28.85ns de dead time por el otro, pulsos simétricos */

PWMCON2bits.IUE = 0;

/* desactiva la actualizacion inmediata del ciclo de trabajo */

PWMCON2bits.ITB = 0;

/* se selecciona modo base de tiempo primaria, mediante ptper */

FCLCON2bits.FLTMOD = 3;    /* desactivadas faltas */

TRGCON2bits.TRGDIV = 0;

/* se fija el disparo de la interrupcion de adc, cada evento de pwm */

TRGCON2bits.TRGSTRT = 0;

/* disparo de la 1ª interrupción desde el 1er evento de pwm */

 PDC2 = 72;         /* fija un ciclo de trabajo inicial, donde el minimo es (dtr2+altdtr2) */

 TRIG2 = 0;

/* retardo de disparo de la interrupcion de adc fijado en 0, desde el evento de pwm */
```

```
}

void CurrentandVoltageMeasurements(void)

{

        /*configuracion del módulo adc del reductor 1 y 2*/


 ADCONbits.FORM = 0;        /* dato de salida del adc como entero (10 bits) */

 ADCONbits.EIE = 0;

 /* interrupcion tras conversion completada en los dos pines del par*/

ADCONbits.ORDER = 0;        /* conversion primero en el pin par */

ADCONbits.SEQSAMP = 0;/* converion simultanea del par de entradas analogicas */

ADCONbits.ADCS = 7;         /* reloj del adc = fadc/8 = 120mhz / 8 = 15mhz,*/

 IFS6bits.ADCP1IF = 0;              /* limpia el flag de interrupcion del adc */

 IPC27bits.ADCP1IP = 5;

 /* fija la prioridad de la interrupcion (0 disabled, 1 la mas baja, 7 la mas alta) */

 IEC6bits.ADCP1IE = 1;              /* habilita interrupcion de adc del par 1 */

ADPCFGbits.PCFG0 = 0;

/*pin como entrada analógica. Medida de corriente del reductor 2 */

ADPCFGbits.PCFG1 = 0;

/*pin como entrada analogica. medida de tension del reductor 2*/

ADPCFGbits.PCFG4 = 0;

/* pin como entrada analogica. medida de tension de entrada a la placa buck/bost */

 ADSTATbits.P0RDY = 0;

/* reset flag dato listo para el par 0 (set por hardware cuando se tiene el dato en
buffer) */

ADCPC0bits.IRQEN0 = 1;
```

```
/* habilitada interrupcion del adc para el lazo del reductor 1 */
ADSTATbits.P1RDY = 0;          /* reset flag dato listo para el par 1 */
ADCPC0bits.IRQEN1 = 1;
 /* habilitada interrupcion del adc para el lazo del reductor 2 */
ADCPC0bits.TRGSRC1 = 5;      /* lectura del par 1 del adc por pwm2 */
ADSTATbits.P2RDY = 0;          /* reset flag dato listo para el par 2 */
ADCPC1bits.IRQEN2 = 0;         /* deshabilitada interrupcion del adc para vin */
ADCPC1bits.TRGSRC2 = 4;      /* lectura del par 2 del adc por pwm1 */
        /*configuracion de la interrupcion por boton de la placa starter board*/
RPINR0bits.INT1R = 8;
/*pin14 (rp8) mapeable, como entrada a la interrupcion externa 1*/
IFS1bits.INT1IF = 0;               /*reset flag interrupcion externa 1*/
IPC5bits.INT1IP = 4;
/*prioridad de la interrupcion externa 1(0 disabled, 1 la mas baja, 7 la mas alta) */
IEC1bits.INT1IE = 1;
  /*habilitada interrupcion provocada por interrupcion externa 1 */
}
void Buck2VoltageLoop(void)
{
        /*se fijan en memoria x e y, los coeficientes y las variable tpid*/
Buck2VoltagePID.abcCoefficients = Buck2VoltageABC;
/* puntero a la zona de memoria x donde estan los coeficientes de la fdt */
Buck2VoltagePID.control = Buck2Voltage;
/* puntero a la zona de memoria y donde estan los errores y ciclos de trabajo */
```

```
PIDInit(&Buck2VoltagePID);
```

/* se llama a funcion pidinit, se le pasan las zonas de memoria x e y para inicializarlas */

```
if ((PID_BUCK2_A1 == 0x7FFF || PID_BUCK2_A1 == 0x8000) ||
(PID_BUCK2_A2 == 0x7FFF || PID_BUCK2_A2 == 0x8000) ||
(PID_BUCK2_A3 == 0x7FFF || PID_BUCK2_A3 == 0x8000) ||
(PID_BUCK2_B0 == 0x7FFF || PID_BUCK2_B0 == 0x8000) ||
(PID_BUCK2_B1 == 0x7FFF || PID_BUCK2_B1 == 0x8000) ||
(PID_BUCK2_B2 == 0x7FFF || PID_BUCK2_B2 == 0x8000) ||
(PID_BUCK2_B3 == 0x7FFF || PID_BUCK2_B3 == 0x8000))
{
while(1);
```

/* comprobacion de coeficientes en q15, si alguno es q15 entra en bucle infinito */

```
}
/* ubica los coeficientes en sus posiciones de memoria */
Buck2VoltagePID.abcCoefficients[0] = PID_BUCK2_B0;
Buck2VoltagePID.abcCoefficients[1] = PID_BUCK2_B1;
Buck2VoltagePID.abcCoefficients[2] = PID_BUCK2_B2;
Buck2VoltagePID.abcCoefficients[3] = PID_BUCK2_B3;
Buck2VoltagePID.abcCoefficients[4] = PID_BUCK2_A1;
Buck2VoltagePID.abcCoefficients[5] = PID_BUCK2_A2;
Buck2VoltagePID.abcCoefficients[6] = PID_BUCK2_A3;
Buck2VoltagePID.measuredOutput = 0;
/* se inicializa el campo measuredoutput de la variable tpid*/
}
```

```
void Buck2SoftStartRoutine(void)
{
/* rutina de arranque suave. Hace crecer el valor de referencia hasta alcanzar el valor final.
este proceso se hace en 50 pasos de 1 ms cada uno, llegando a la referencia final en 50 ms */
while
(Buck2VoltagePID.controlReference <= PID_BUCK2_VOLTAGE_REFERENCE)
{
        Delay_ms(1);
        Buck2VoltagePID.controlReference+=BUCK2_SOFTSTART_INCREMENT;
}
Buck2VoltagePID.controlReference = PID_BUCK2_VOLTAGE_REFERENCE;
/* se introduce en la variable tpid, la referencia actual*/
reference1=PID_BUCK2_VOLTAGE_REFERENCE;
/* dos referencia para hacer escalones de ref. */
reference2=PID_BUCK2_VOLTAGE_REFERENCE2;
}
void Delay_ms (unsigned int delay)
/*funcion para hacer retardos de (unsigned int delay) ms. 1ms en este programa */
{
TimerInterruptCount = 0;      /* reset del contador de interrupciones del timer */
PR1 = 0x9C40;                 /* (1ms / 25ns) = 40,000 = 0x9C40 */
IPC0bits.T1IP = 4; /* se fija la prioridad de la interrupcion menor que la del adc */
IEC0bits.T1IE = 1;            /* se habilita la interrupcion de timer1 */
T1CONbits.TON = 1;            /* arranca la cuenta del timer1 */
```

```
while (TimerInterruptCount < delay);
/* espera a que el numero de interrupciones llegue a (unsigned int delay). 1ms */
T1CONbits.TON = 0;                    /* detiene la cuenta del timer */
}
```

11.1.2 Isr.c

```
#include "p33FJ16GS502.h"
#include "dsp.h"
unsigned int TimerInterruptCount = 0;
unsigned int flag = 0;
unsigned int reference1;
unsigned int reference2;
tPID Buck2VoltagePID;
/*se ejecuta esta subrutina al desbordar el timer1*/
void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt()
{
  TimerInterruptCount ++;
/* se incrementa el contador del retardo para la rampa de referencia */
 IFS0bits.T1IF = 0;                /* reset del flag de esta interrupcion */
}
/*se ejecuata esta subrutina cuando se pulsa el boton de la placa started board*/
void __attribute__((__interrupt__, no_auto_psv)) _INT1Interrupt()
{
        if (flag == 0)                    /*si esta como referecnia el "reference1"...*/
        {
                flag=1;                   /*se cambia el flag al cambiar de referencia*/
Buck2VoltagePID.controlReference=reference2;
        /*se pone la nueva referencia en la variable tpid*/
IFS1bits.INT1IF = 0;                      /*reset del flag de esta interrupcion*/
        }
```

```
        else                    /*si esta como referecnia el "reference2"...*/
        {
            flag=0;             /*se cambia el flag al cambiar de referencia*/
        /*se pone la nueva referencia en la variable tpid*/
Buck2VoltagePID.controlReference=reference1;
IFS1bits.INT1IF = 0;            /*reset del flag de esta interrupcion*/
        }
}
```

11.1.3 Main.c

```
#include "p33FJ16GS502.h"
#include "Functions.h"
_FOSC(FCKSM_CSDCMD & OSCIOFNC_ON)        /* configuration bits de mplab*/
_FWDT(FWDTEN_OFF)
_FPOR(FPWRT_PWR128))
_FICD(ICS_PGD2 & JTAGEN_OFF)
#define INPUTUNDERVOLTAGE 335
/* fija un umbral de vin minimo en 6v→ */
/*2.2k/(10k+2.2k)*6V = 1.081V →1.081/3.3*1023 = 335 */
#define INPUTOVERVOLTAGE 894
/* umbral de vin maximo en 16v→ 2.2k/ (10k+2.2k)*16 = 2.884V  2.884/3.3*1023 = 894 */
int main(void)
{
int InputVoltage;
                        /*configuracion del oscilador para funcionar a 40mhz*/
                        /*fosc= fin*m/(n1*n2), fcy=fosc/2*/
                        /*fosc= 7.37*(43)/(2*2)=80mhz para fosc, y fcy = 40mhz */
PLLFBD=41;                          /* M = PLLFBD + 2 */
CLKDIVbits.PLLPOST=0;               /* N1 = 2 */
CLKDIVbits.PLLPRE=0;                /* N2 = 2 */
__builtin_write_OSCCONH(0x01);      /* seleccion de frc w/ pll */
__builtin_write_OSCCONL(0x01);      /* enable switch */
```

```
while(OSCCONbits.COSC != 0b001); /* espera a switch del oscilador, frc w/ pll */
/*sincroniza el dspic con el depurador mplab icd2 y el programa mplab ide*/
while(OSCCONbits.LOCK != 1);

ACLKCONbits.FRCSEL = 1;          /* frc (x16) para pll auxiliar */
ACLKCONbits.SELACLK = 1;         /* oscilador auxiliar para el reloj de pwm y adc */
ACLKCONbits.APSTSCLR = 7;        /* reloj auxiliar dividido entre 1 */
ACLKCONbits.ENAPLL = 1;          /* habilitado pll auxiliar */
while(ACLKCONbits.APLLCK != 1);  /* espera al pll auxiliar */
PTPER = 3205;                    /*se fija la frecuencia de conmutacion en 300khz*/
                                 /* ptper = ((1 / 300khz) / 1.04ns) = 3205 */
 TRISAbits.TRISA3 = 0;           /*llamadas a funciones de inicializacion de perifericos*/
Buck2Drive();                    /* funcion que configura el reductor 2 */
 CurrentandVoltageMeasurements(); /* funcion para configurar adc */
Buck2VoltageLoop();              /* funcion que inicializa la variable tpid del redutor 2 */

ADCONbits.ADON = 1;                          /* adc activado */
PTCONbits.PTEN = 1;                          /* pwm activado */
        /*a partir de aqui pueden darse las interrupcion del adc y saltar a su subrutina*/
Buck2SoftStartRoutine();         /* se inicializa el arranque suave del reductor 2 */

 while(1)                /*bucle infinito a la espera de interrupciones del adc para hacer */

/*recalculos del valor de control (que se van haciendo constantemente)*/
{
if (ADSTATbits.P2RDY ==1)                /*si ha habido lectura...*/
                {
        InputVoltage = ADCBUF4;  /* lectura del valor leido de tension de entrada */
        ADSTATbits.P2RDY = 0;            /* reset del flag de lectura en buffer */
                }
if((InputVoltage<=INPUTUNDERVOLTAGE)||(InputVoltage>=INPUTOVERVOLTAGE))
                                         /*si vin esta fuera de rango...*/
                {
```

```
        IOCON2bits.OVRENH = 1;          /* detiene el pwmh del reductor */
        IOCON2bits.OVRENL = 1;          /* detiene el pwml del reductor */
                }
}                                       /*bucle infinito*/
}
```

11.1.4 ISRasm.s

```
        .equ __33FJ16GS502, 1
        .include "p33FJ16GS502.inc"
; maximo ciclo de trabajo del  reductor 2 para el control por tension
        .equ Buck2MaxDC, 3000
; minimo ciclo de trabajo del  reductor 2 para el control por tension
        .equ Buck2MinDC, 72
; valores para poder posicionar o buscar en memoria los cambios
        .equ    offsetabcCoefficients, 0
        .equ    offsetcontrol, 2
        .equ    offsetcontrolOutput, 4
        .equ    offsetmeasuredOutput, 6
        .equ    offsetcontrolReference, 8
        .data
        .text
; definicion de la subrutina de atencion a la interrupcion del adc1 (reductor 2)
        .global __ADCP1Interrupt
; inicio de la subrutina de atencion a la interrupcion de adc1
__ADCP1Interrupt:
; se guarda en la pila lo que hay en los registros de trabajo w0 y w1 para luego recuperarlos
        push w0
        push w1
; se pone el pin3 del puerto a '1'. se usa para medir el tiempo de recalculo del nuevo d.
        bset PORTA, #3
; se guarda en w0, el valor de la posicion de memoria donde empieza la variable tpid
        mov #_Buck2VoltagePID, w0
; se guarda en w3 los 10 bits de la lectura de tension de salida del reductor2
```

```
        mov ADCBUF3, w1
; se escala la lectura en x2^5. se desplazan los 10 bits 5 posiciones
; para tenerlo en la parte mas significativa del registro de 16 bits, w1
        sl  w1, #5, w1
; la medida escalada se guarda en [w0+6], que es
; su sitio de memoria dentro de la variable tpid
        mov w1, [w0+#offsetmeasuredOutput]
; se llama a la funcion _pid, que calcula el nuevo valor de control
        Call _PID
; el nuevo valor que se guardo en [w0+4] en la variable tpid y que
; el valor de control calculado en tpid, se mete en w1
        mov.w [w0+#offsetcontrolOutput], w1 ;
; se escala en /2^1
        asr w1, #1, w0
; comprueba si el valor de control es menor que el minimo
        mov.w #Buck2MinDC, w1
        cpsgt w0, w1
; si es menor que el minimo, se pone el "buck2mindc", 72
        mov.w w1, w0
; comprueba si el valor de control es mayor que el maximo
        mov.w #Buck2MaxDC, w1
        cpslt w0, w1
; si es mayor que el maximo, se pone el "buck2maxdc", 3000
        mov.w w1, w0
; el nuevo valor de d, se mete en el registro que modula en pwm
        mov.w w0, PDC2
; se limpian los flags de conversion e interrupcion del adc
        bclr    ADSTAT,#0           ; Clear Pair 0 conversion status bit
        bclr    IFS6, #15           ; Clear Pair 0 Interrupt Flag
; se pone el pin3 del puerto a '0'.es posible ver con osciloscopio cuanto tardo este proceso
        bclr    PORTA, #3
; se recupera de la pila lo que habia en los registros que han sido usados
        pop w1
        pop w0
```

; se sale de la subrutina de interrupcion

retfie

11.1.5 PID.s

; se añade el fichero dspcommon.inc

```
.nolist
.include "dspcommon.inc"
.list
```

; valores para poder posicionar o buscar en memoria los cambios

```
.equ offsetabcCoefficients, 0
.equ offsetcontrol, 2
.equ offsetcontrolOutput, 4
.equ offsetmeasuredOutput, 6
.equ offsetcontrolReference, 8

.section .libdsp, code          ; se añade la libreria dsp
```

; definicion de la funcion _pid

```
.global _PID
```

; inicio de la funcion

```
_PID:
```

; se guarda en la pila lo que hay en distintos registros de trabajo para luego recuperarlos

```
push.s
push w4
push w5
push w8
push w10
push CORCON
```

; se toman los valores de los campos de la variable tpid en los siguientes resitros de trabajo

;w8 <--- puntero a la primera posicion de memoria con coneficientes de la fdt (memoria x)

```
mov [w0 + #offsetabcCoefficients], w8
```

;w10 <----- puntero a la primera posicion de memoria con error o d (memoria y)

```
mov [w0 + #offsetcontrol], w10
```

;w1 <----- valor d antes de hacer el nuevo calculo

```
        mov [w0 + #offsetcontrolOutput], w1
;w2 <----- valor actual de la variable a controlar (vout del reductor 2)
        mov [w0 + #offsetmeasuredOutput], w2
;w3 <----- valor actual de la referencia a alcanzar
        mov [w0 + #offsetcontrolReference], w3
;recolocacion de las posiciones de error en sus nuevas posiciones
; dejando e[n] listo para pisarlo con el valor nuevo de error.
        mov [w10 + #+4],w11
        mov     w11, [w10 + #+6] ;     E[n-3]<==E[n-2]
        mov [w10 + #+2],w11
        mov     w11, [w10 + #+4] ;     E[n-2]<==E[n-1]
        mov [w10],w11
        mov w11, [w10 + #+2] ;         E[n-1]<==E[n]
                                       ;  ([W10])-->E[n]<== LIBRE
; acumulador a<-----w3(referencia), a = tpid.controlreference
        lac     w3, a
; acumulador b<-----w2(valor actual de la variable a controlar), b = tpid.measuredoutput
        lac     w2, b
; acumulador a<-----referencia-valor actual, error [n]
        sub     a
; se redondea y se mete en [w10], e[n] (se redondea, porque acc 40 bits y [w10], 16 bits)
        sac.r   a, [w10]
; se limpia el acumulador a, y se prepara en w4 y w5
        clr     a, [w8]+=2, w4, [w10]+=2, w5          ; w4 = B0, w5 = E[n]
; se limpia b
        clr       b
; se multiplica y se prepara la siguiente multiplicacion
; a = b0 * e[n]
; w4 = b1, w5 = e[n-1]
        mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
; a = b0 * e[n] + b1 * e[n-1]
; w4 = b2, w5 = e[n-2]
        mac     w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2]
```

```
; w4 = b3, w5 = e[n-3]
        mac    w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3]
; w4 = a1, w5 = d[n-1]
        mac    w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
; se mete d[n-1] en b
        lac            w1, b
;a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + d[n-1]
        add            a
;a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + 2 * d[n-1]
        add            a
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + 2 * d[n-1] + a1 * d[n-1]
; en este caso el coeficiente a1, tiene parte entera '2' y parte fraccionaria a1
; w4 = a2, w5 = d[n-2]
        mac            w4*w5, a, [w8]+=2, w4, [w10]+=2, w5
; se mete d[n-2] en b
        lac            w5,    b
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + (2+a1) * d[n-1] + d[n-2]
; el este caso el coeficiente va restando
        sub            a
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + (2 + a1) * d[n-1] + (1+a2) * d[n-2]
; en este caso el coeficiente a2, tiene parte entera '1' y parte fraccionaria a2
; w4 = a3, w5 = d[n-3]
        mac            w4*w5, a, [w8]+=2, w4, [w10]-=4, w5
; a = b0 * e[n] + b1 * e[n-1] + b2 * e[n-2] + b3 * e[n-3] + (2 + a1) * d[n-1] + (1+a2) * d[n-2] + a3 * d[n-3]
        mac            w4*w5, a,
; a += a3 * d[n-3]
; se tiene en a, el valor de la ecuacion en diferencias del regulador tipo3
; se redondea el valor de d que hay en acc a (40 bits) y se mete la parte mas significativa en w1(16 bits)
        sac.r    a, w1
; recolocacion de las posiciones de d en sus nuevas posiciones
        mov            [w10 + #+4],w11
        mov            w11, [w10 + #+6]      ;       d[n-3]<==d[n-2,anterior]
        mov            [w10 + #+2],w11
```

```
mov     w11, [w10 + #+4]    ;   d[n-2]<==d[n-1,anterior]
mov     [w10],w11
mov     w11, [w10 + #+2]    ;   d[n-1]<==d[n,anterior]
mov     w1,[w10]
mov     w1, [w0 + #offsetcontrolOutput]
```

;d[n]<==w1(d[n] alculado ahora) ; se recupera de la pila lo que habia en los registros que han sido usados

```
pop    CORCON
       pop    w10
       pop    w8
       pop    w5
       pop    w4
       pop.s
       return
```

; definicion de la funcion _pidinit

```
    .global _PIDInit
```

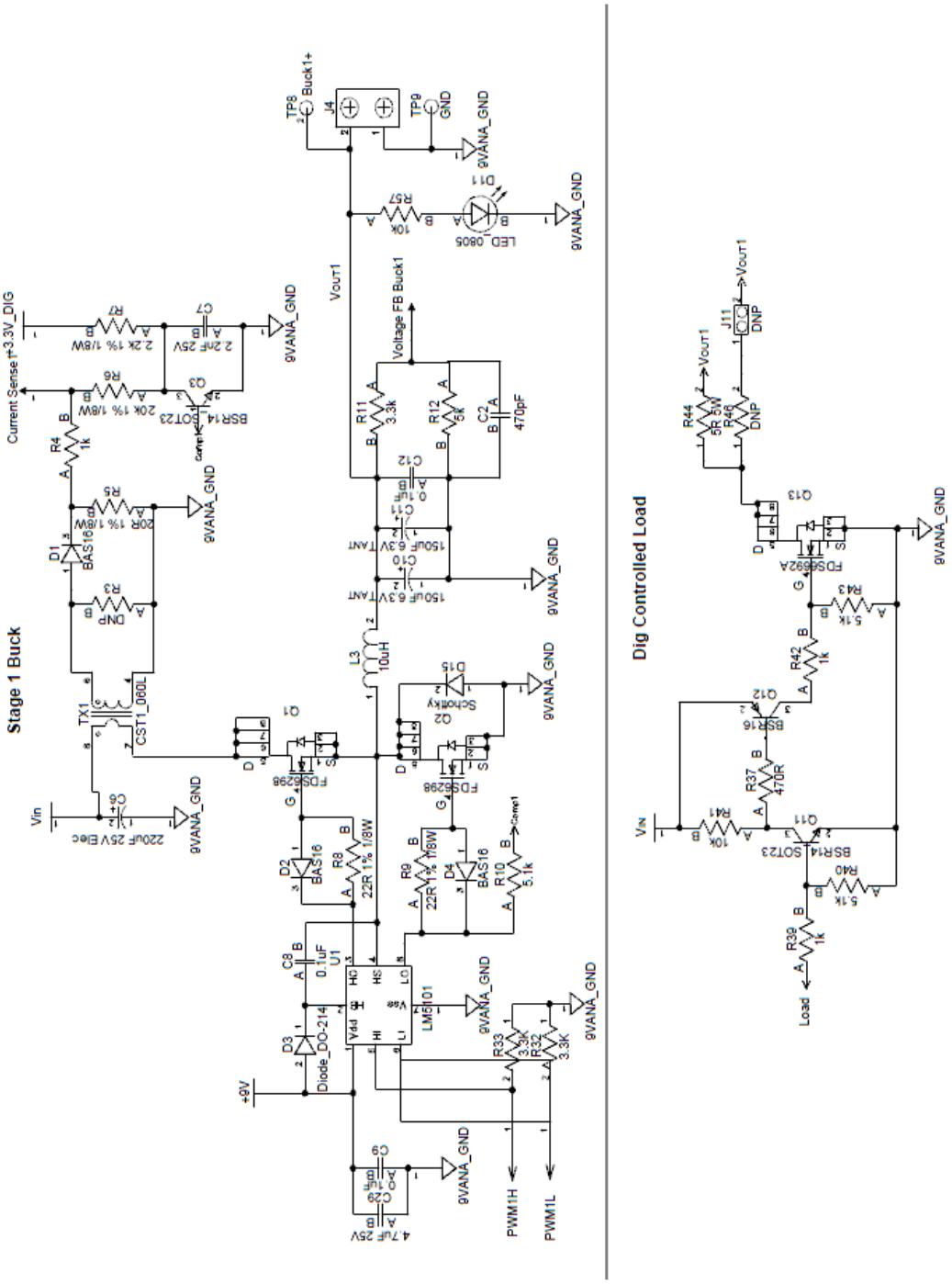
; inicio de la funcion

```
_PIDInit:
```

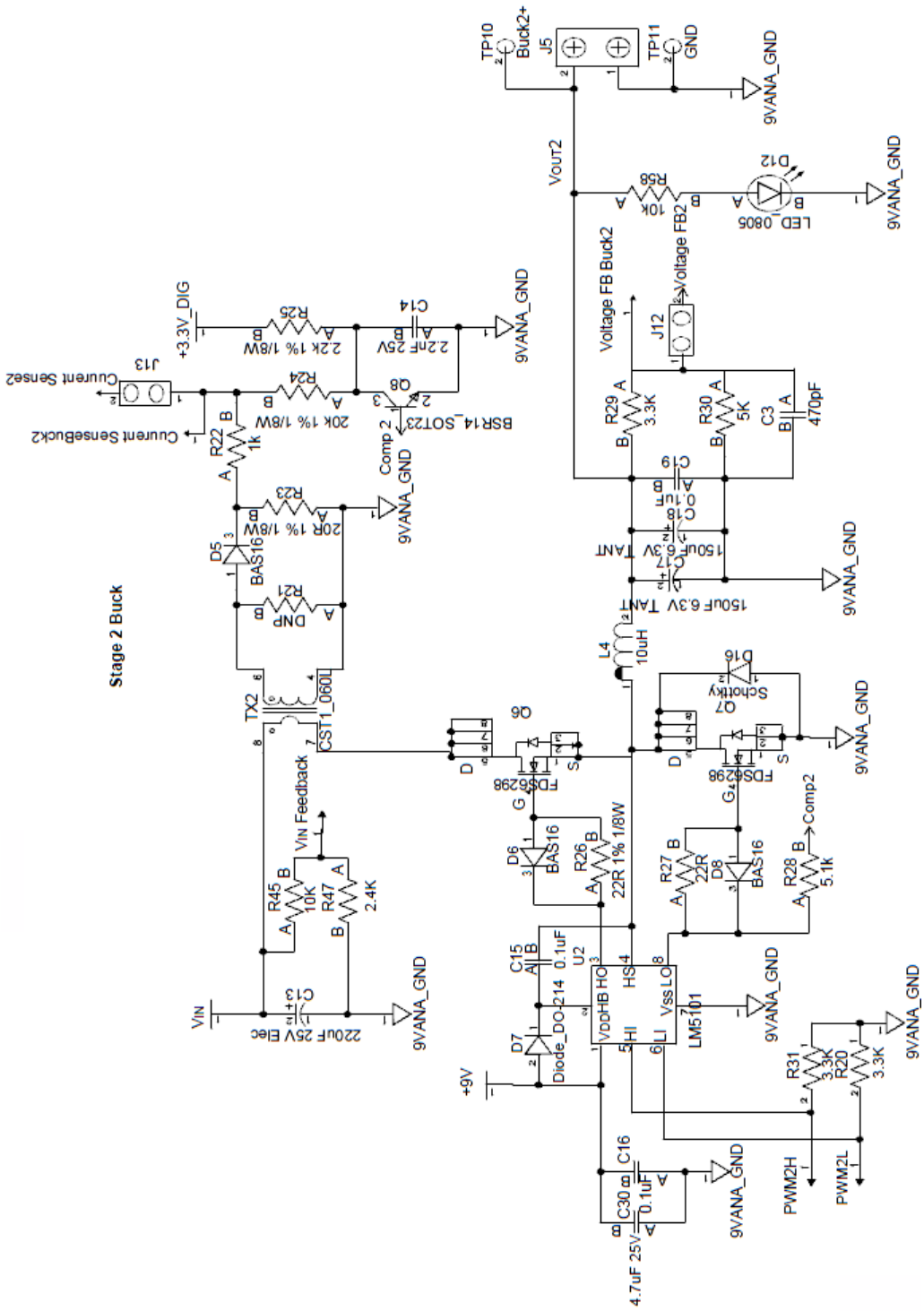
; se resetean las posiciones de memoria donde iran los errores y los d.

```
       push   w0
       add    #offsetcontrolOutput, w0
       clr    [w0]
       pop    w0
       push   w0
       mov    [w0 + #offsetcontrol], w0
       clr    [w0++]          ; d[n]=0
       clr    [w0++]          ; d[n-1] = 0
       clr    [w0++]          ; d[n-2] = 0
       clr    [w0++]          ; E[n] = 0
       clr    [w0++]          ; E[n-1] = 0
       clr    [w0++]          ; E[n-2] = 0
       pop    w0                  return
      .end
```

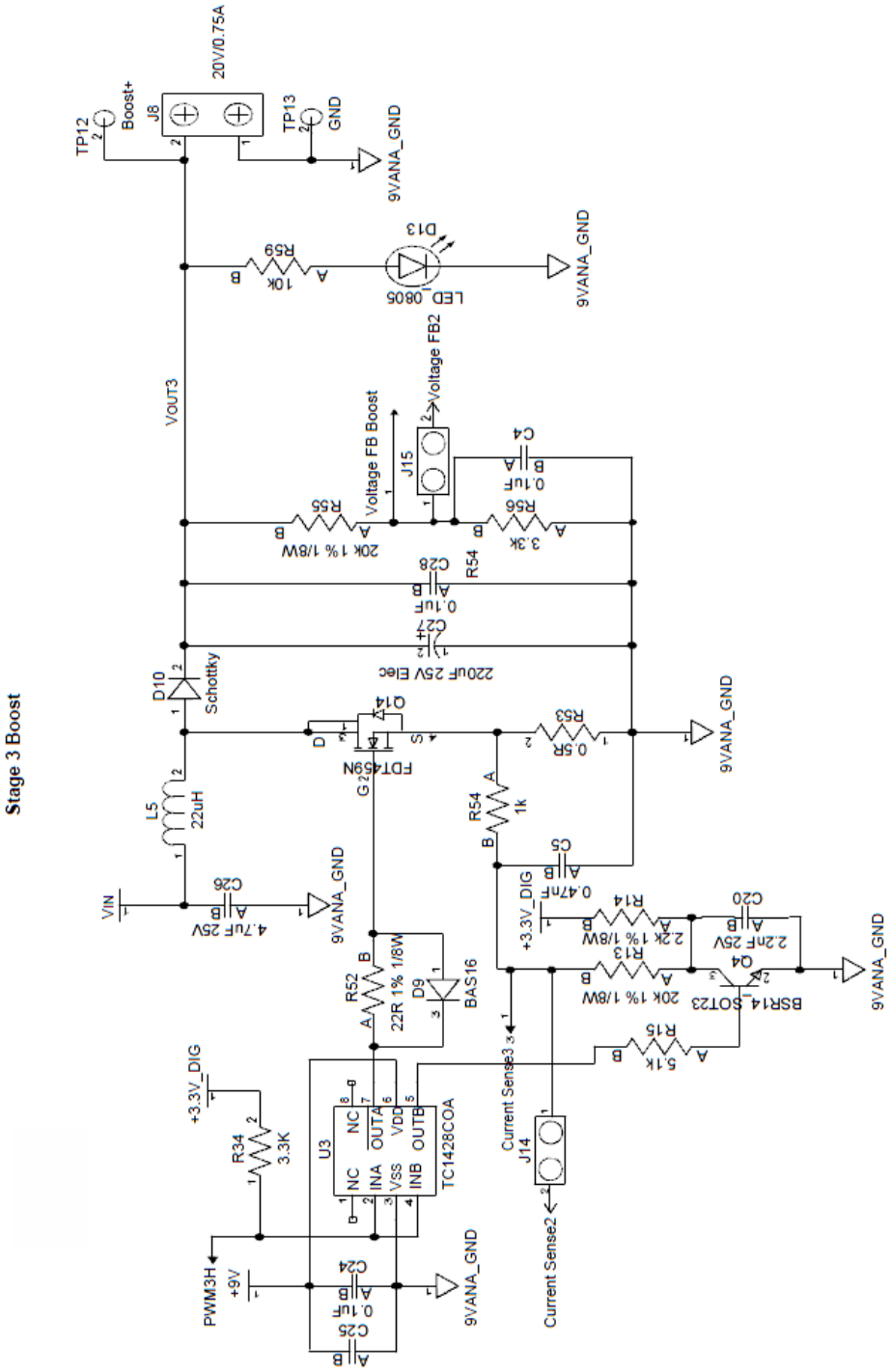
11.2 Esquemáticos de las tarjetas usadas



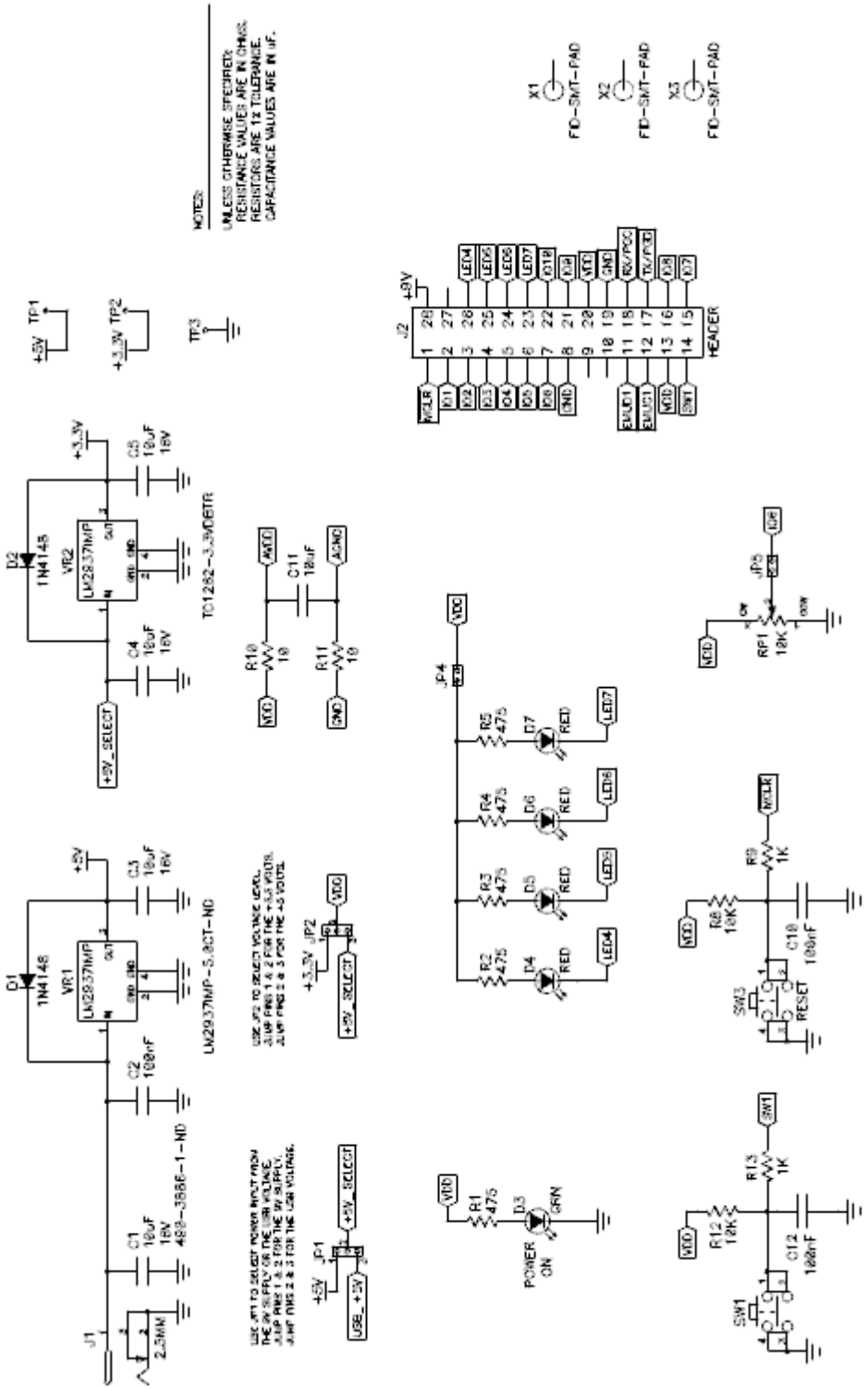
Extraído de [5]



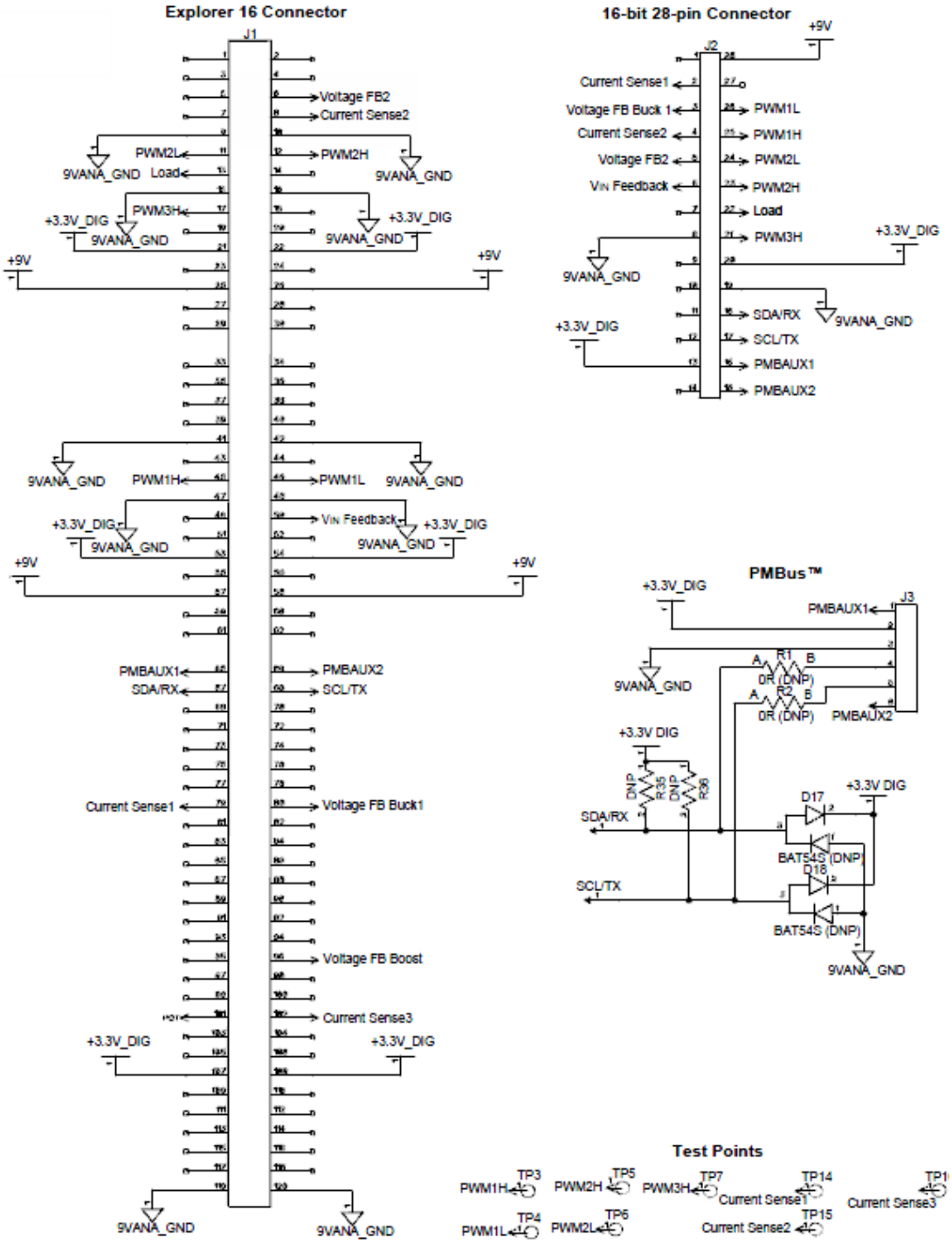
extraído de [5]



extraído de [5]



extraído de [6]



Conexión placa convertidores con placa de control por conector J2,extraído de [6]

12 Presupuesto

En este capítulo se describe la valoración de los costes de realización del proyecto. Para la realización de este proyecto por un lado se hablará de desarrollo de ingeniería y documentación, y por otro de las tarjetas comerciales y aparatos que han sido usados y comprados específicamente para su uso en este proyecto.

Respecto a los aparatos de medida, han sido usados los existentes en el laboratorio y no se añaden a este presupuesto ya que son usados de forma generalizada en el laboratorio, no siendo específicos para este proyecto. Entre estos están los aparatos necesarios para la medida de respuesta en frecuencia que son, los analizadores de ganancia y fase del laboratorio (el HP 4194A y el Venable 3235), el *bode box* y sus respectivas sondas, además de los osciloscopios del laboratorio. Por otro lado para el desarrollo de este proyecto son necesarias las licencias de los software no gratuitos usados para el desarrollo de este proyecto y que son los mismos que se necesitan para repetir o continuar los trabajos realizados. Entre estos están el PSIM con SmartCtrl para el diseño de los lazos y el MatLab para discretizar los controles obtenidos.

No se ha fabricado ningún componente o prototipo, por lo que no hay coste de fabricación ya que se han usado tarjetas comerciales para el desarrollo y medida experimental de los trabajos realizados en este proyecto.

12.1 Costes de desarrollo

En esta parte se especifican los costes del estudio, diseño, implementación y medida de lo desarrollado para alcanzar los objetivos del proyecto con un coste estimado de 40 €/hora. Y también los costes de documentación del trabajo realizado con un valor de 15 €/hora

ACTIVIDAD	*€/hora*	*Nºhoras*	*COSTE*
Desarrollo de ingeniería	-	-	-
Diseño, implementación y medidas experimentales	*40*	*600*	*24000€*
Documentación	-	-	-
Redacción del documento técnico y recopilación de datos experimentales	*15*	*50*	*750€*
TOTAL			*24750€*

Tabla 12-1 Costes de desarrollo

12.2 Costes de material específico

Para el desarrollo de este proyecto, ha sido necesario comprar distintos componentes de Microchip. Estos materiales han sido específicamente adquiridos para este proyecto por lo que se incluyen en el presupuesto. Además para poder realizar el proceso de diseño e implementación de reguladores digitales en dsPIC con ayuda de la herramienta CAD, SmartCtrl, estos compontes son necesarios para obtener los resultados experimentales detallados en este proyecto.

Componente	*Referencia Microchip*	*Unidades*	*Precio/unidad*	*precio*
Buck/Boost PICtail™ Plus Daughter Card	*AC164133*	*2*	*62,93 €/ud*	*125,86 €*
16-Bit 28P Starter Demo Board	*DM300027*	*2*	*55,94 €/ud*	*111,88 €*
MPLAB ICD 3 In-Circuit Debugger	*DV164005*	*1*	*132,86 €/ud*	*132,86 €*
dsPIC33FJ16GS502	*-*	*10*	*2,97 €/ud*	*29,7 €*
TOTAL			*254,7 €/ud*	*400,3 €*

Tabla 12-2 Costes de componentes

12.3 Presupuesto del proyecto

De la suma de los apartados anteriores:

TIPO DE COSTE	*COSTE*
Costes de desarrollo	*24750 €*
Costes de componentes	*400 €*
Total antes de I.V.A	*25150 €*
IVA (16%)	*4024 €*
TOTAL	*29174 €*

Tabla 12-3 Presupuesto del proyecto realizado

13 Bibliografía

[1] "Fundamentals of power electronics"
2nd Edition, Kluwer 2001
Robert. W. Ericsson, Dragan Maksimovic

[2] Ingeniería de control moderna
4ª Edición, Pearson-Prentice-Hall
Katsuhiko Ogata

[3] "Testing power sources for stability"
Venable technical paper #1
H. Dean Venable

[4] "dsPIC33FJ06GS101/X02 and dsPIC33FJ16GSX02/X04 data sheet, High-Performance, 16 bit Digital Signal Controllers"
2009 Microchip Technology Inc.

[5] "Buck/Boost Converter PICtail™ Plus Daughter Board, User´s guide"
2010 Microchip Technology Inc.

[6] "16-Bit 28-Pin Starter Development Board User´s guide"
2008 Microchip technology Inc.

Referencias generales no citadas específicamente en el texto del documento

[7] "dsPIC30F/33F Programmer´s Reference Manual, High-Performance Digital Signal Controllers"
2005 Microchip Technology Inc

[8] "Switch Mode Power Supply (SMPS) Topologies (Part I-II)"
2007 Microchip Technology
Mohammad Kamil, Antonio Bersani

[9] "Specify Gain And Phase Margins On All Your Loops"
Venable technical paper #2
H. Dean Venable

[10] "Optimum Feddback Amplifier Design For Control Systems"
Venable technical paper #3
H. Dean Venable

[11] "Herramienta para el diseño automático de reguladores digitales basados en FPGA para convertidores de potencia conmutados"
PFC, UC3M
C. Serrano, P. Zumel